GDPR Metadata Indexing Optimization

Shrief Hussien
Advisor: Dr. Dimitrios Stavrakakis
Chair of Computer Systems
https://dse.in.tum.de/

1/4 – 30/9

Outline

- Background & Motivation
- Design
- Implementation
- Evaluation
- Summary & Future Work

General Data Protection Regulation (GDPR)

- Widespread abuse of personal data
- GDPR grants rights and assigns responsibilities
- GDPR compliance is challenging
  - Metadata explosion
    - e.g., purposes, objections, user, origin, sharing, TTL
  - Logging is expensive
- GDPR introduces new workloads and query patterns
  - e.g., get(key) -> get(all_keys_for_user_A), get(all_keys_with_purpose_A)
  - timely deletion (expiration)

Goal: Build a middleware layer to enable efficient indexing and querying over GDPR-related metadata in key-value stores (e.g., RocksDB, Redis, memcached)

State-of-the-art

- Existing work focuses primarily on compliance enforcement [1,2]
- KV stores (e.g., Redis, RocksDB) lack native support for secondary indexing
- Current solutions are tightly coupled to specific databases (e.g., PostgreSQL) [1,2]

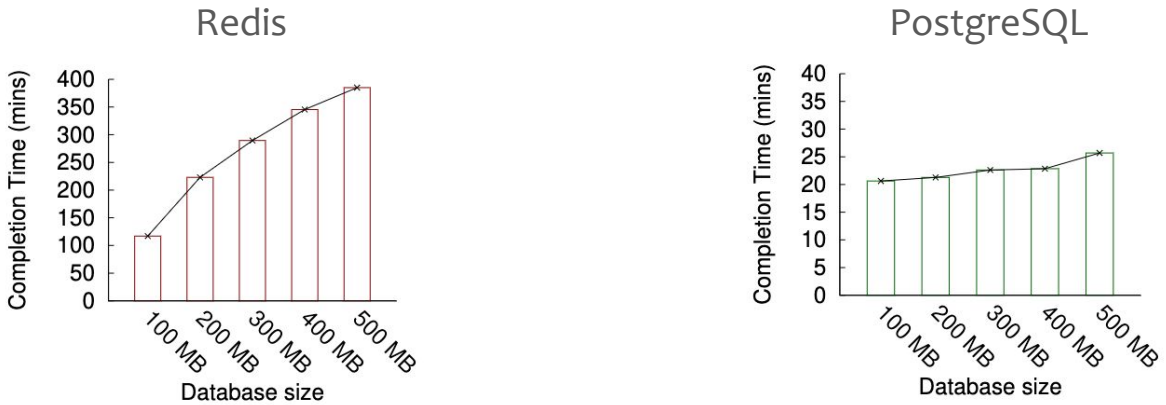
These limitations result in significant performance degradation!

[1] Shastri et al., "Understanding and Benchmarking the Impact of GDPR on Database Systems," PVLDB, 2020.
[2] Shah et al. "Analyzing the Impact of GDPR on Storage Systems," HotStorage, 2019.

State-of-the-art examples

- Redis performs poorly on GDPR-related queries (e.g., 6 hours with 500MB of data size)
- PostgreSQL performance worsens moderately thanks to metadata indices

Completion time for 10K GDPR-related queries



[1] Shastri et al., "Understanding and Benchmarking the Impact of GDPR on Database Systems," PVLDB, 2020.

Problem statement

How can we design *performant* index structures for GDPR metadata that enable **low-latency** metadata queries, minimize **memory footprint**, and preserve system **scalability**?

Outline

- ~~Background & Motivation~~
- Design
- Implementation
- Evaluation
- Summary & Future Work

System: GDPR Metadata Indexing Optimization

Evaluate different indexing strategies to optimize GDPR query performance for Key-value store

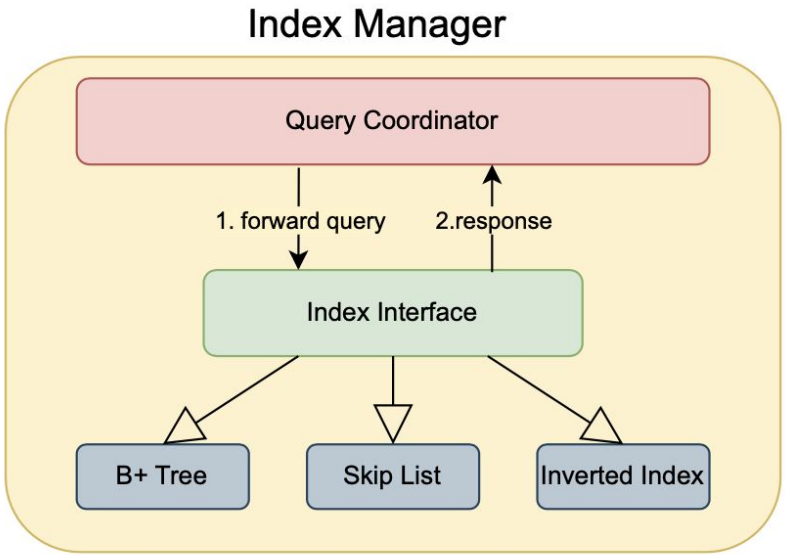
System design goals:

- Scalability
- Consistency
- Extensibility

Backend Agnostic

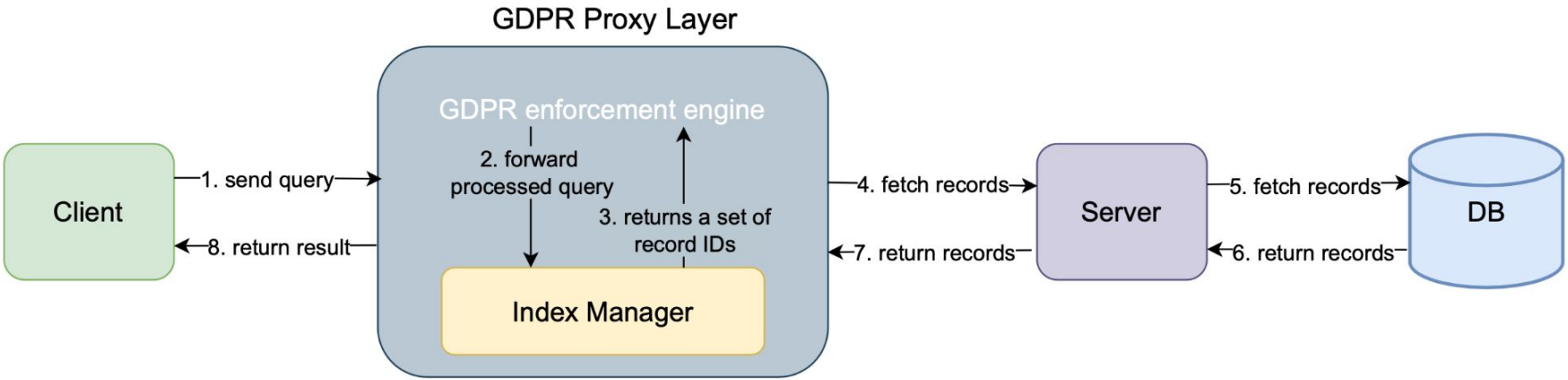
System overview

- Query coordinator receives request from the proxy layer.
- It determines which index or a combination of indices to access.
- The set of values are returned to the proxy layer.
  - key: 12, values: {2, 5, 10}



System Workflow

- Query is routed to the Index Manager.
- A list of matching keys is returned.
- Corresponding values are fetched from the key-value store.



System overview

Supported Queries:

- Find all records for user X –> Subject Index (B+ tree)
- List all records collected for purpose Y –> Purpose Index (Inverted Index)
- Delete all expired records –> Retention Index (Skip List)

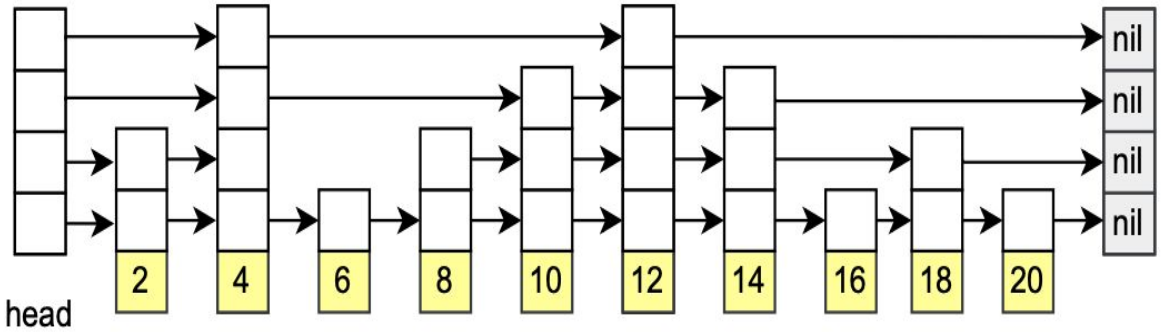
Outline

- ~~Background & Motivation~~
- ~~Design~~
- Implementation
- Evaluation
- Summary & Future Work

Implementation: Retention Index

Enforce data retention policies based on TTL values

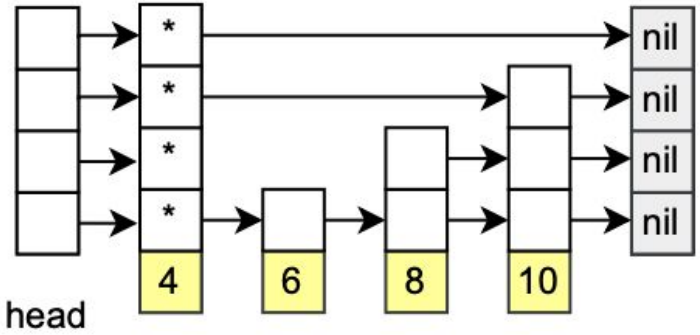
- Ordered by timestamp
- lock-free



Implementation: Retention Index

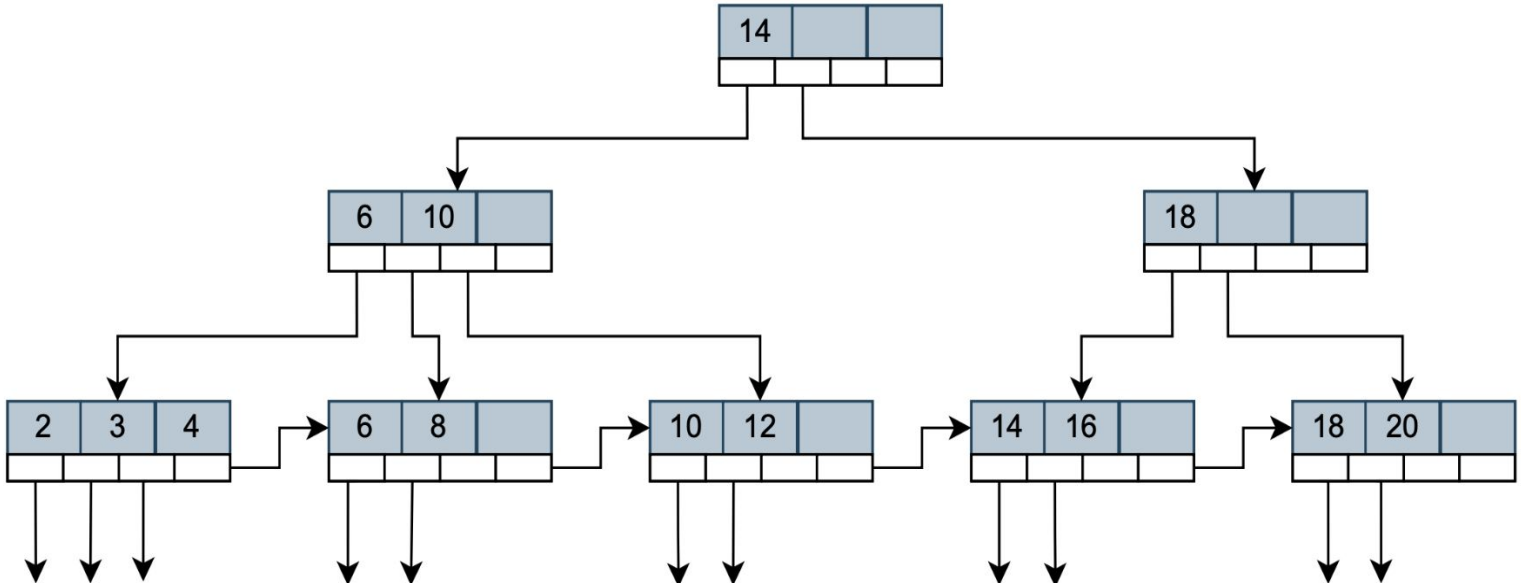
Lock-free skip list:

- Marking the LSB
- CAS atomic operation



Implementation: Subject Index

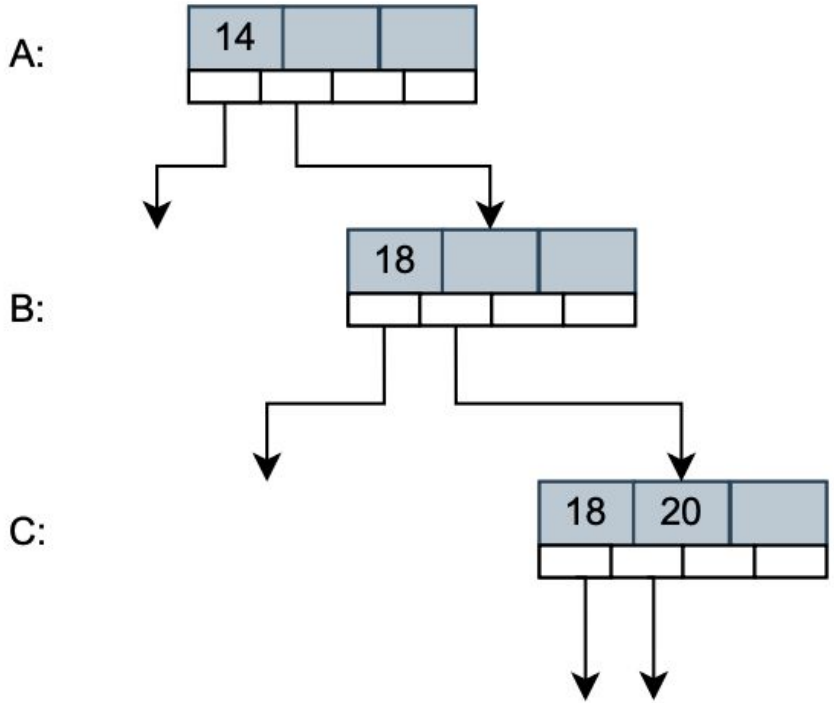
Support efficient lookups and range scans of records grouped by user identifiers



Implementation: Subject Index

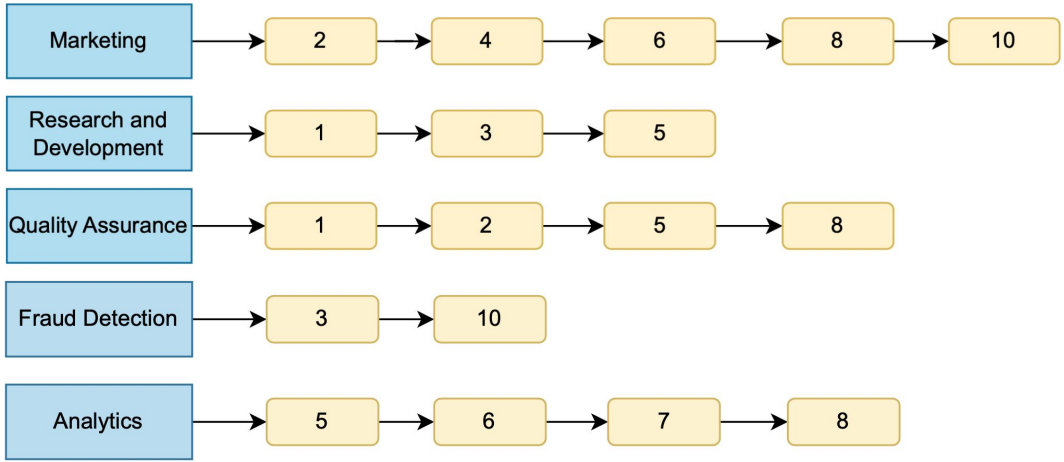
Lock Coupling

1. lock node A
2. access node A
3. lock node B
4. unlock node A
5. access node B
6. lock node C
7. unlock node B
8. access node C
9. unlock node C



Implementation: Purpose Index

Support efficient lookups from a purpose identifier to a set of associated records IDs



Implementation: Purpose Index

- Achieve scalability
- Reduce lock contention

Apply sharding!

Outline

- ~~Background & Motivation~~
- ~~Design~~
- ~~Implementation~~
- Evaluation
- Summary & Future Work

Evaluation

YCSB Workloads:

- Workload A (Update heavy): 50% reads and 50% updates.
- Workload B (Read mostly): 95% reads and 5% updates.
- Workload C (Read only): 100% reads.
- Workload D (Read latest): 95% reads, 5% inserts, focusing on recently added records.
- Workload E (Short ranges): 95% short range scans and 5% inserts.
- Workload F (Read–modify–write): reads a record, modifies it, and writes it back.

Evaluation

Experimental setup

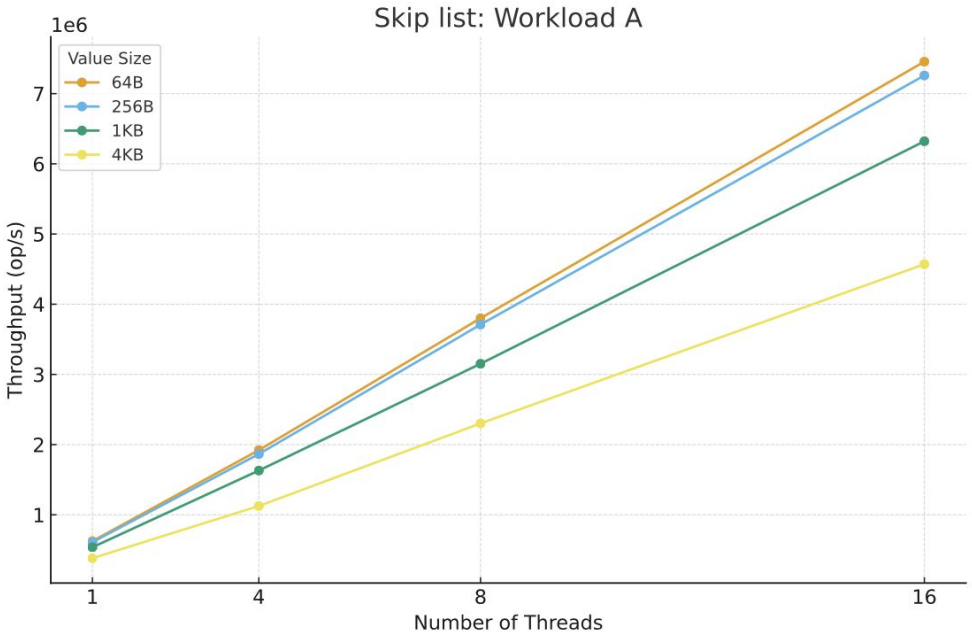
- Worker threads: 1 to 16
- Value size: 64B to 4KB
- Key size: 64B

Metrics

- Throughput (ops/sec)
- Memory consumption over time during inserts.

Evaluation: Retention Index

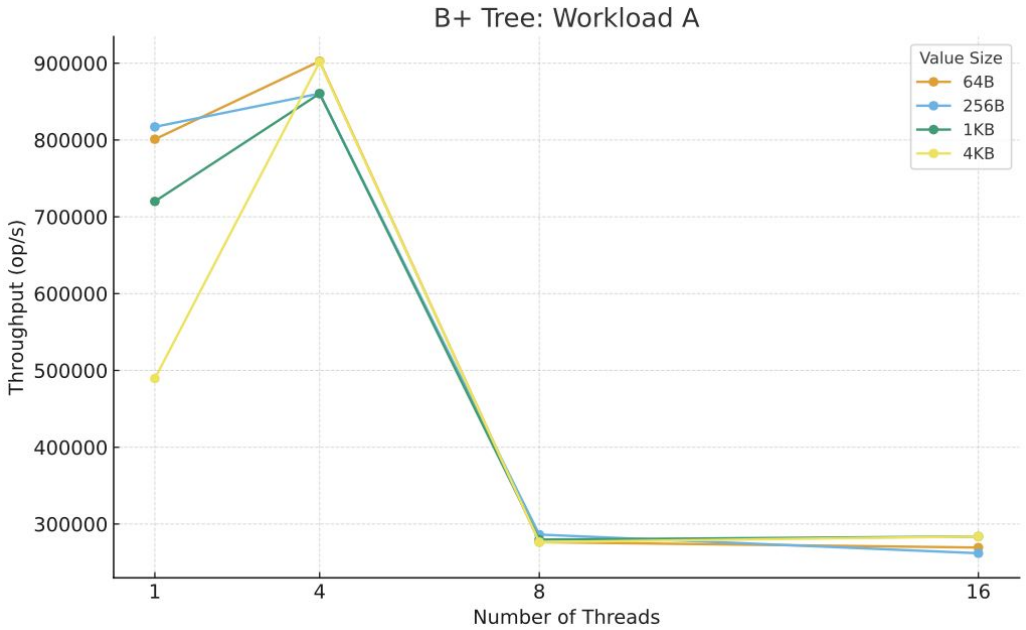
Workload A, **10M** ops, **50%** reads / **50%** updates



Proportional growth under update heavy workloads

Evaluation: Subject Index

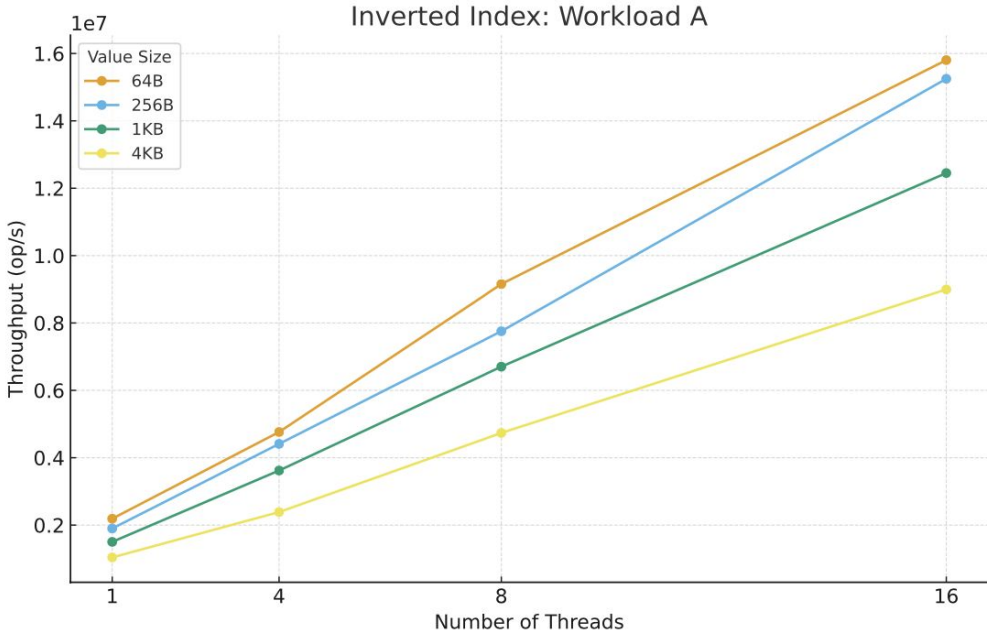
Workload A, **10M** ops, **50%** reads / **50%** updates



Contention on upper-level nodes restricts throughput as concurrency increases

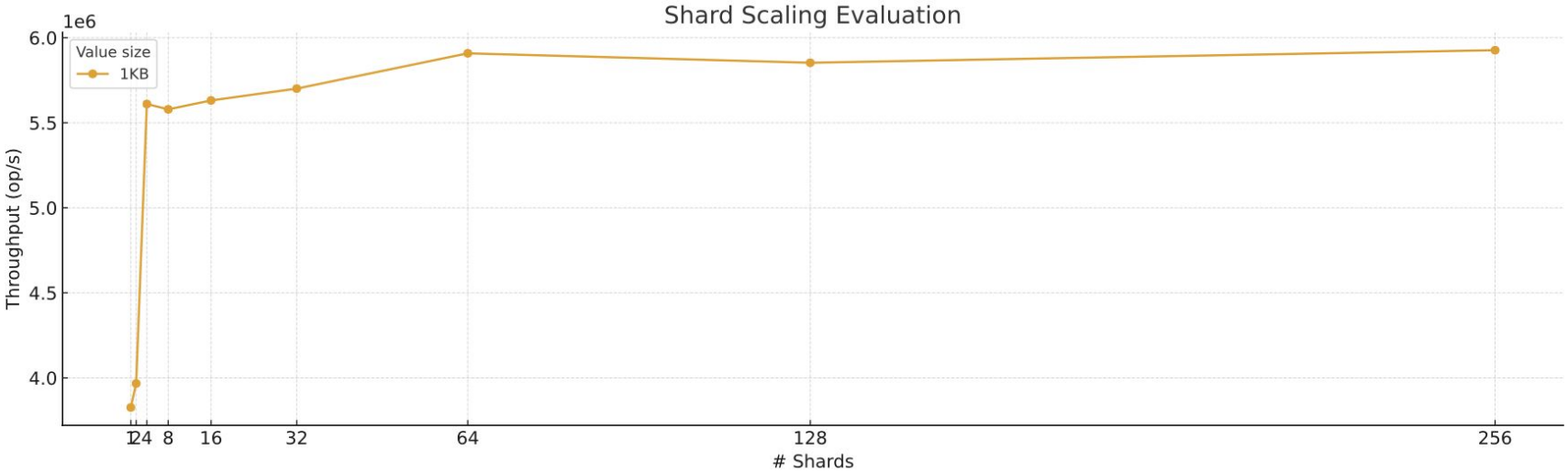
Evaluation: Purpose Index

Workload A, **10M** ops, **50%** reads / **50%** updates



Linear growth and stability under update-intensive workloads

Evaluation: Purpose Index



Outline

- ~~Background & Motivation~~
- ~~Design~~
- ~~Implementation~~
- ~~Evaluation~~
- Summary & Future Work

Summary & Future Work

Key-value stores are not designed to handle GDPR workloads efficiently

- Implemented an extensible **indexing layer**.
- Evaluated the **effectiveness** of different index structures for **GDPR tasks**.

Future work:

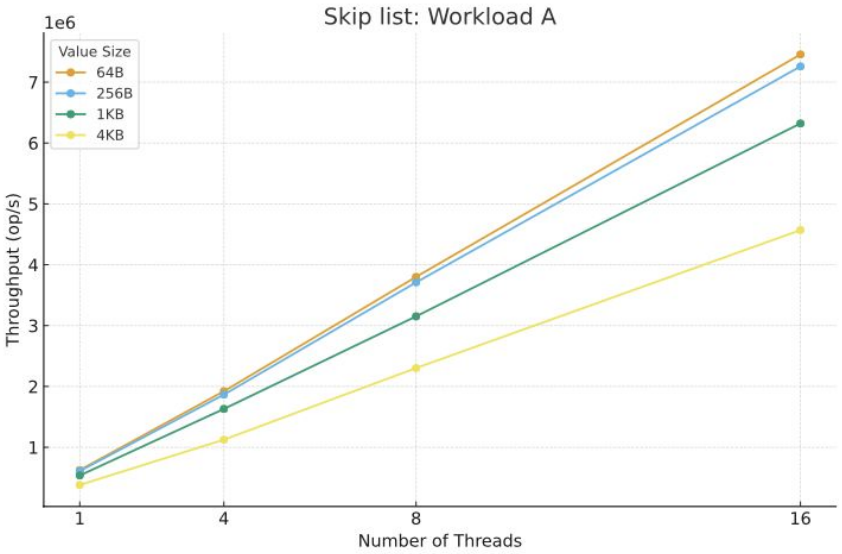
- Extend the index layer to support additional index structures.
- Integrate multi-attribute and temporal/interval indexes for more complex queries.
- Evaluate the system on workloads with GDPR-specific metadata fields to validate real-world performance.

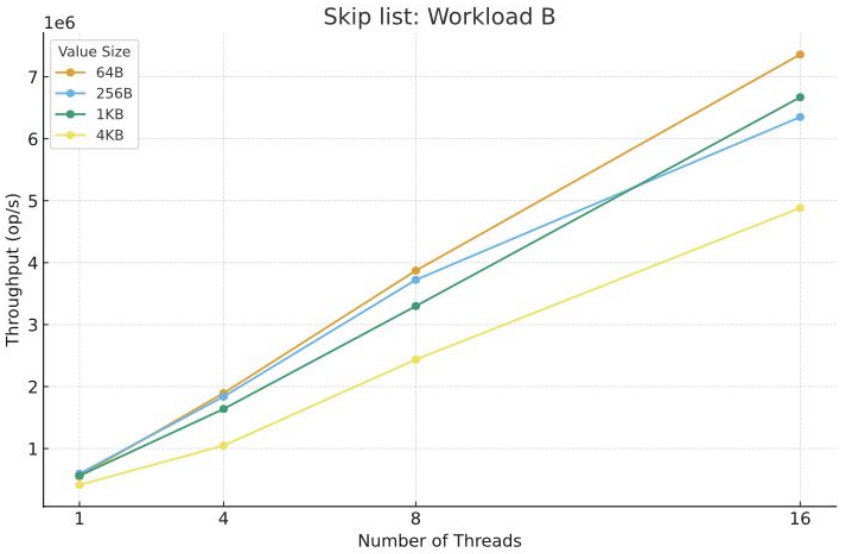
Thank You!

Questions?

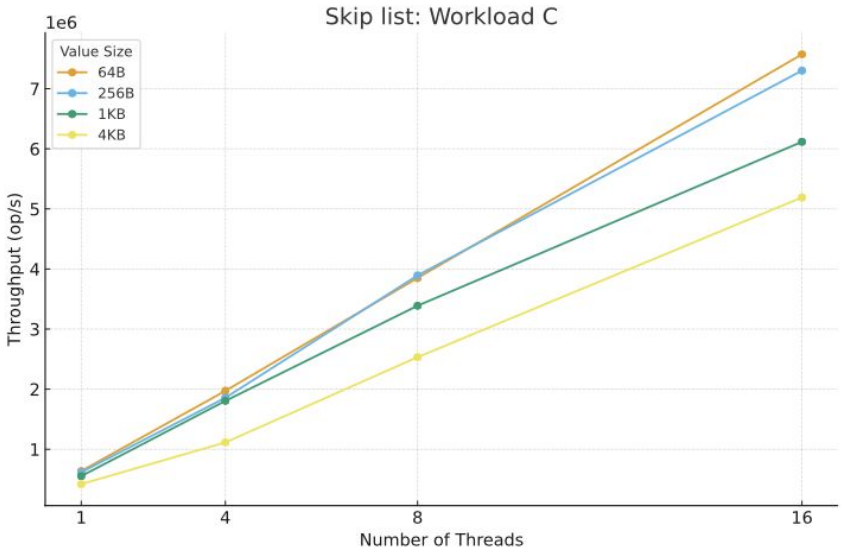
Backup Slides

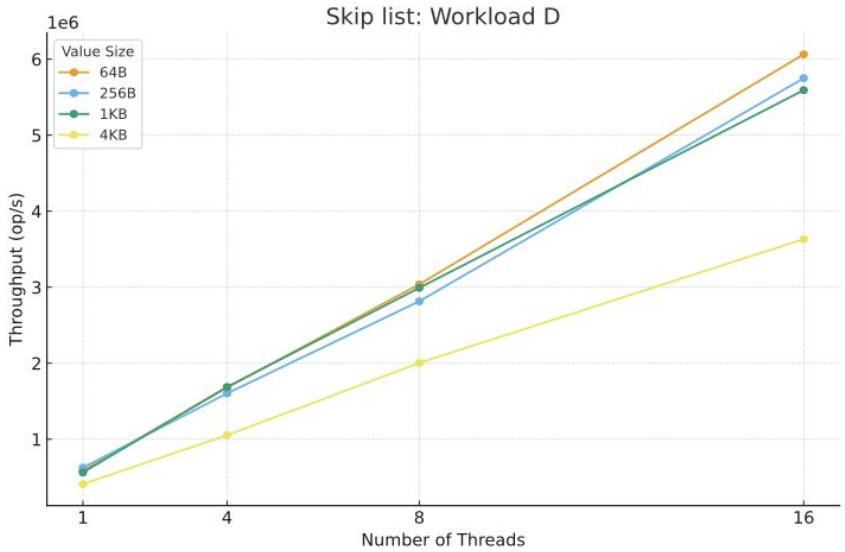
Evaluation: Retention Index



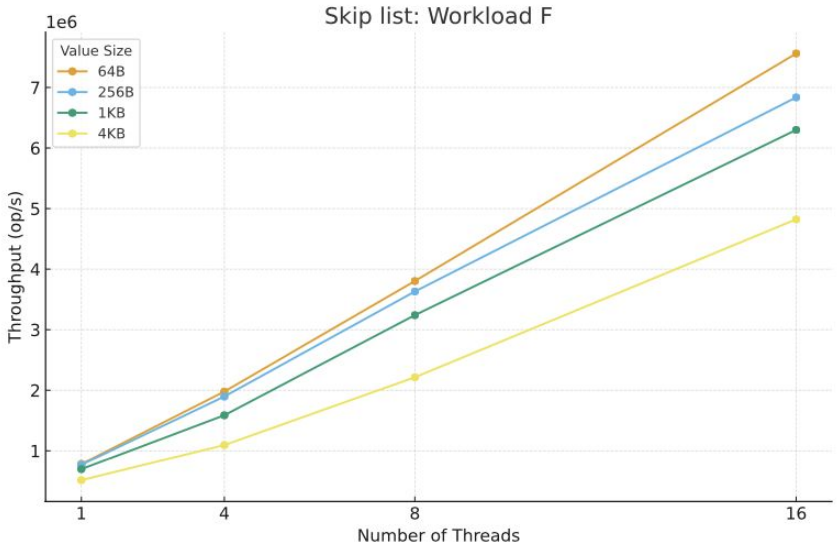


Evaluation: Retention Index

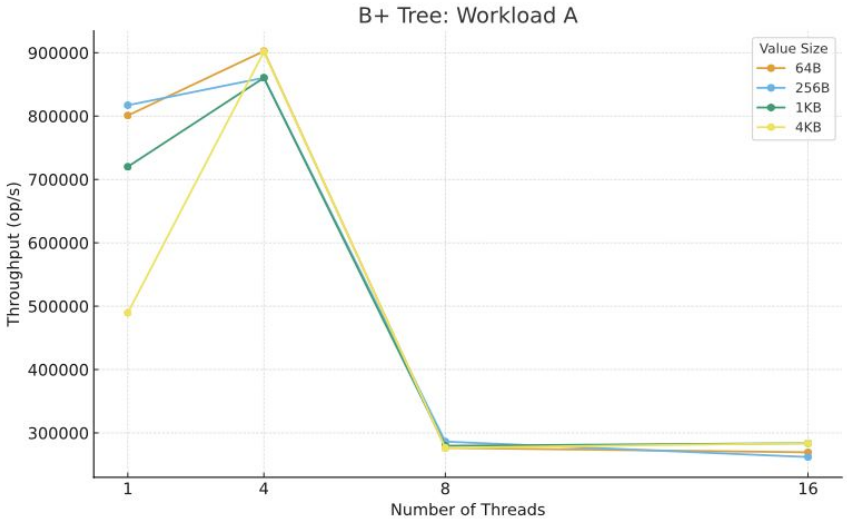


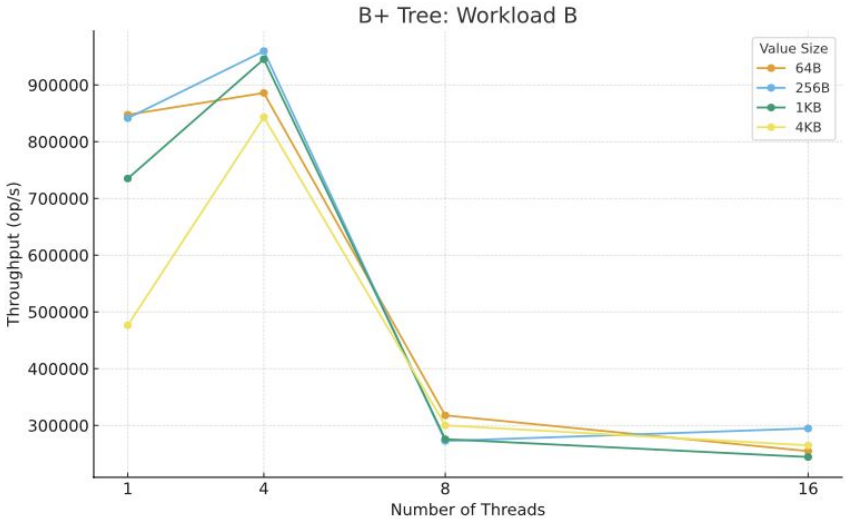


Evaluation: Retention Index

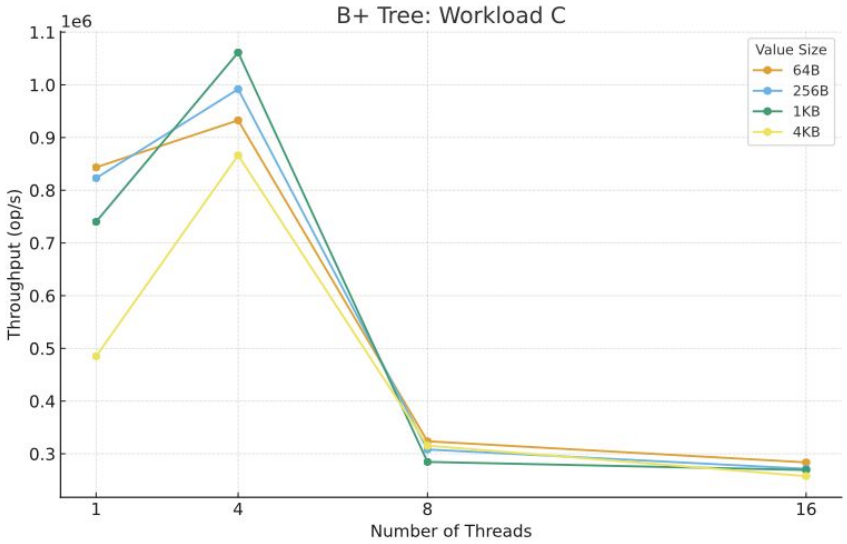


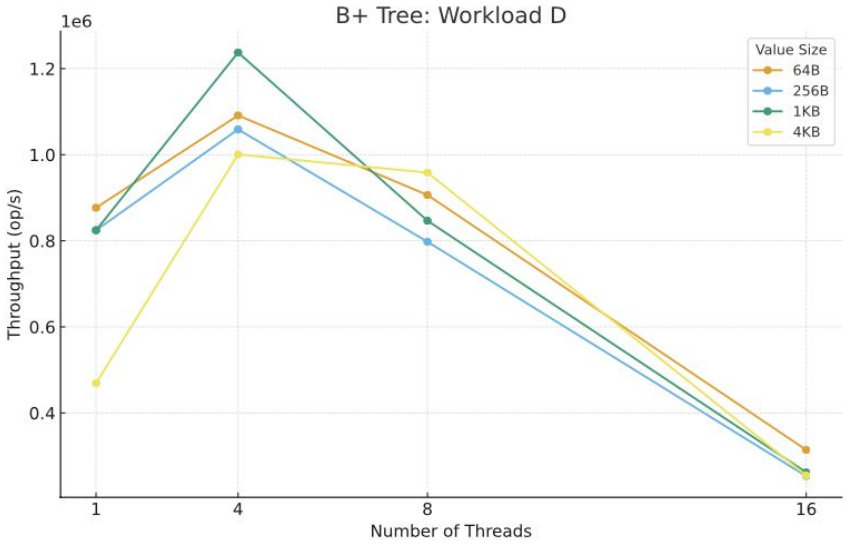
Evaluation: Subject Index



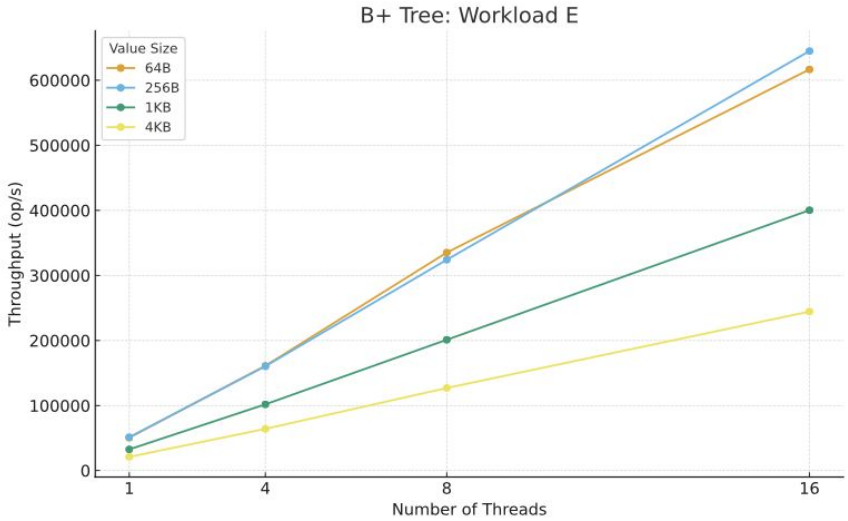


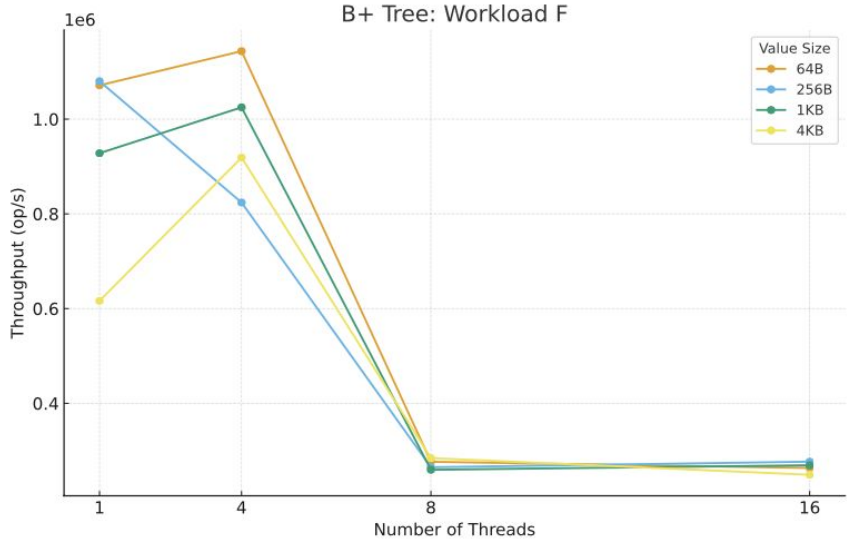
Evaluation: Subject Index



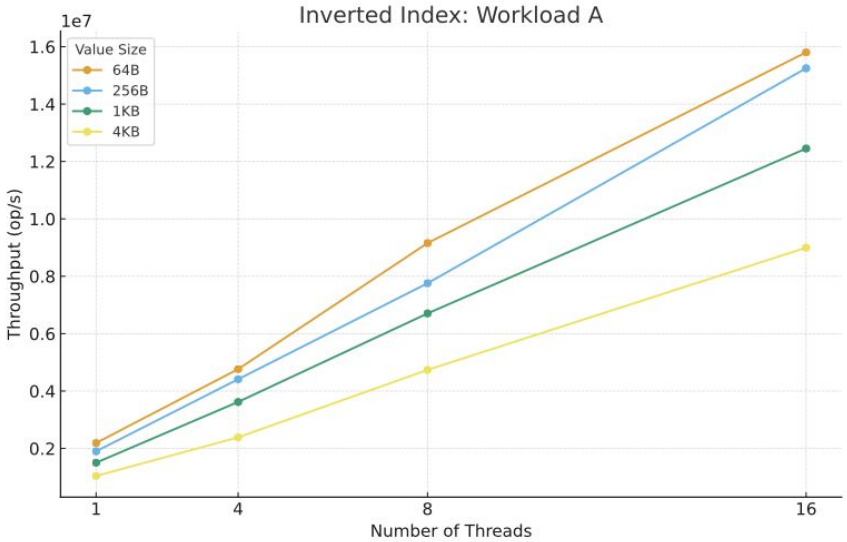


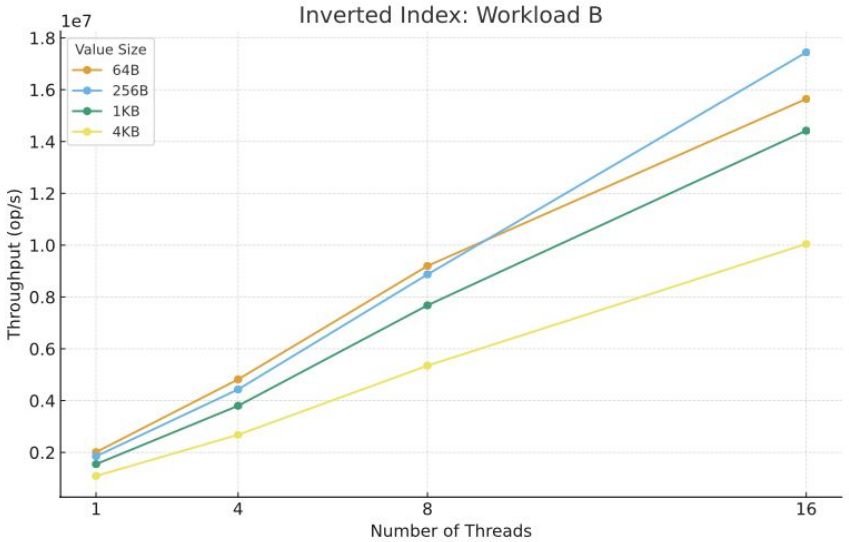
Evaluation: Subject Index



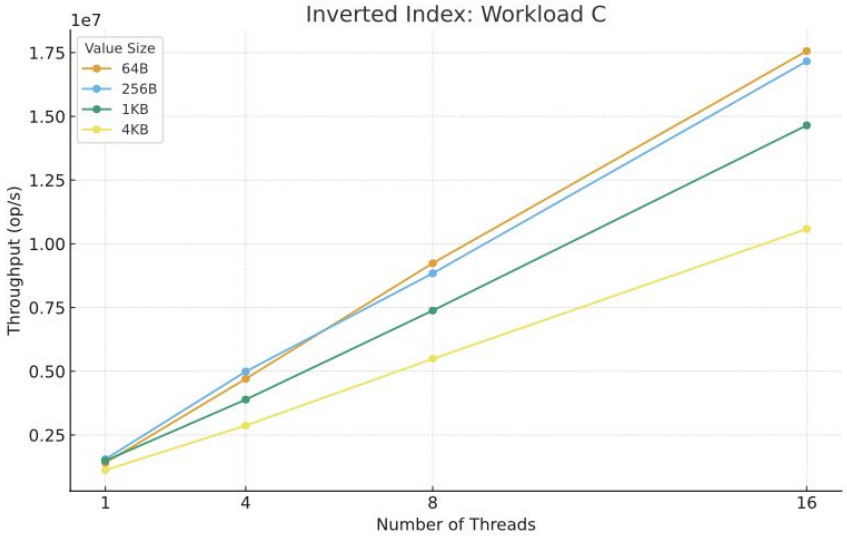


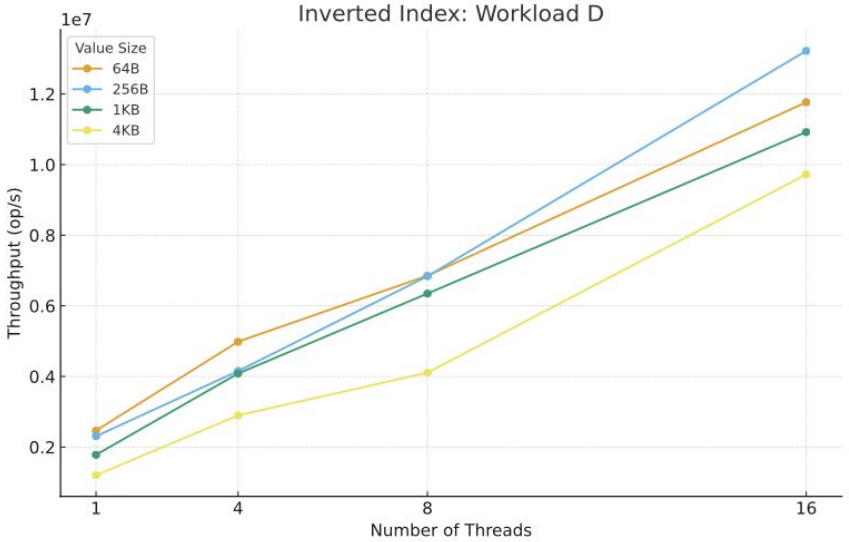
Evaluation: Purpose Index



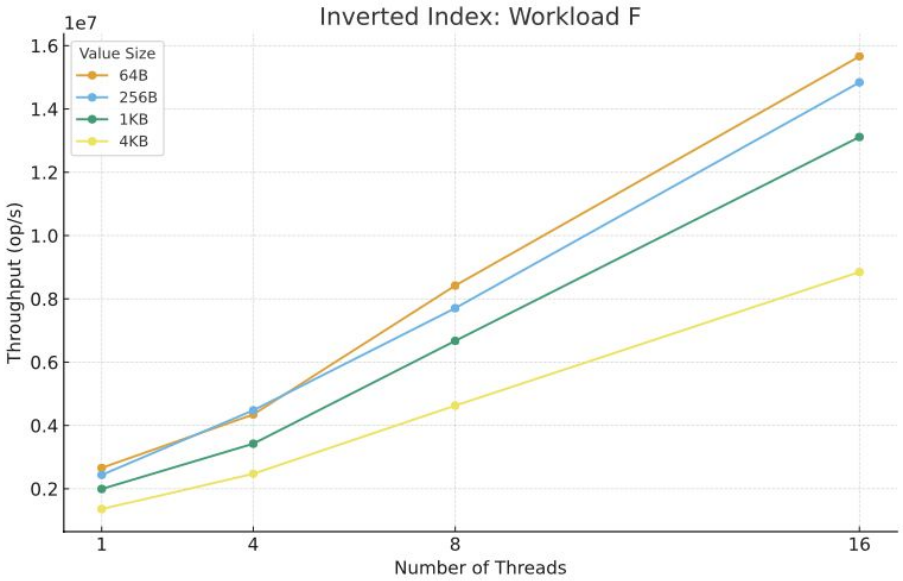


Evaluation: Purpose Index

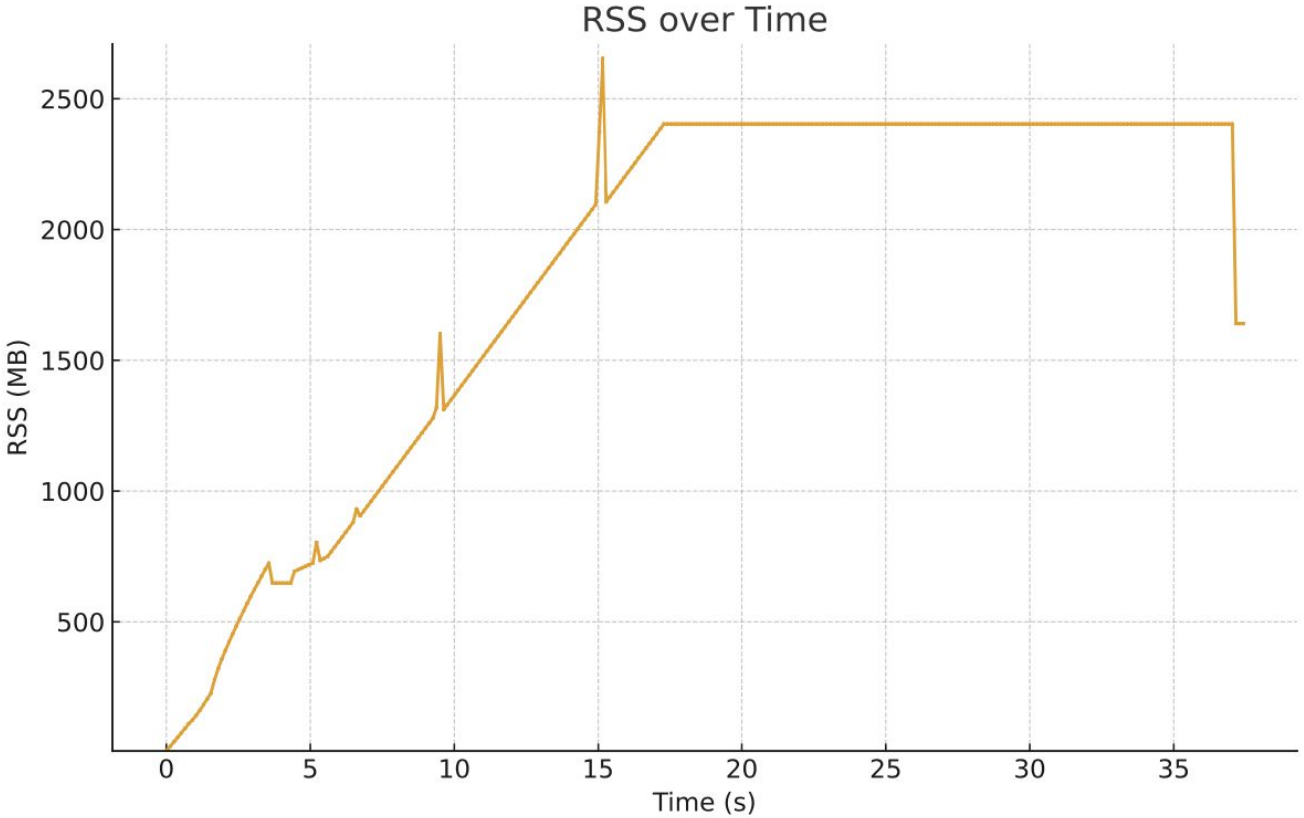




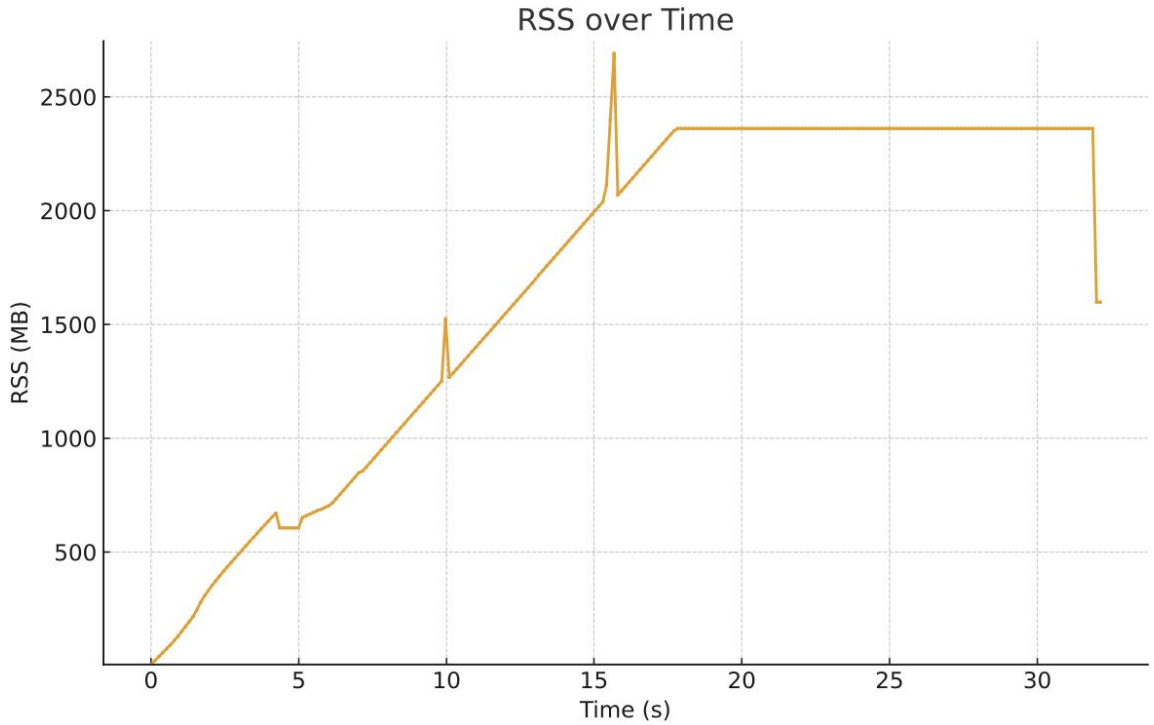
Evaluation: Purpose Index



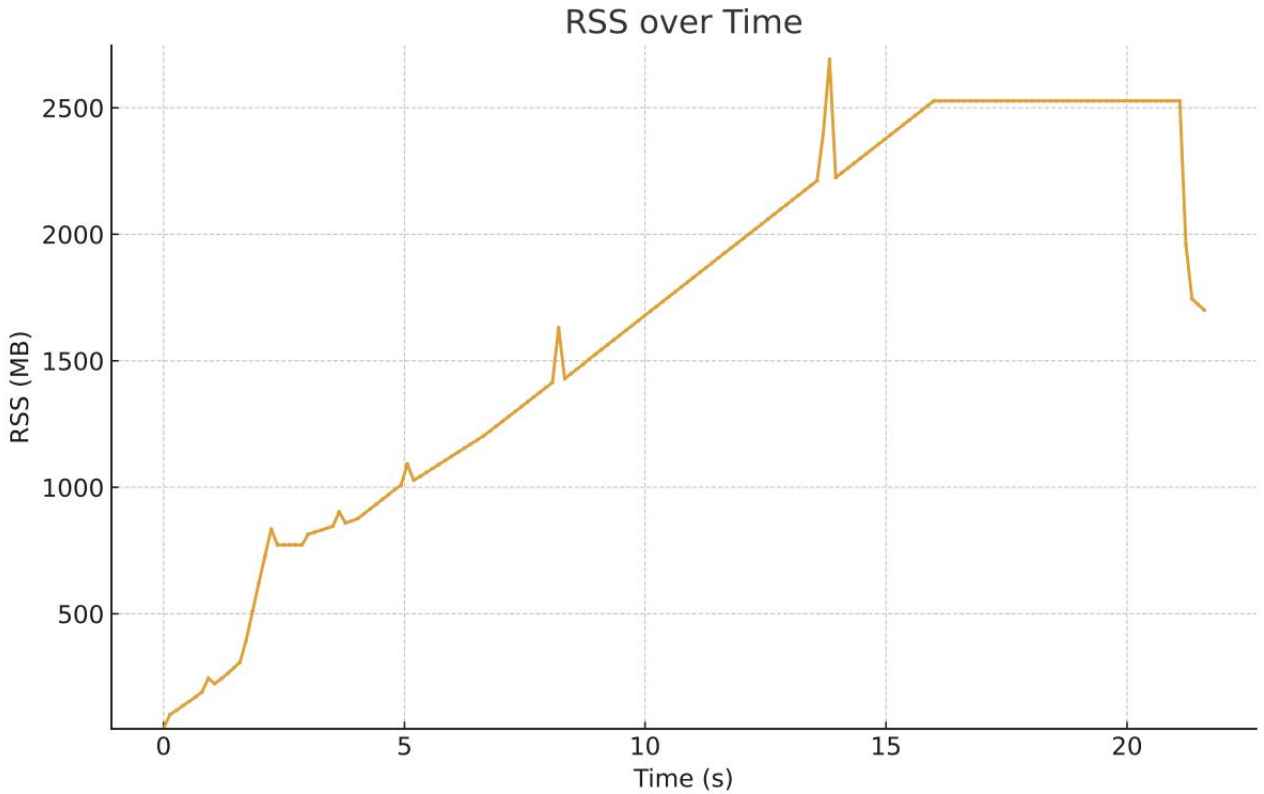
Evaluation: Retention Index



Evaluation: Subject Index



Evaluation: Purpose Index



References

[1] Shastri et al., “Understanding and Benchmarking the Impact of GDPR on Database Systems,” PVLDB, 2020.

[2] Shah et al. “Analyzing the Impact of GDPR on Storage Systems,” HotStorage, 2019.