HEDAS: Secure and Efficient Distributed OLAP using Fully Homomorphic Encryption

Yu Tian, Tianxiang Shen, Qi Hu, Wei Chen, Heming Cui, and Ji Qi*

tianyuk@cs.hku.hk

Sep. 19, 2024

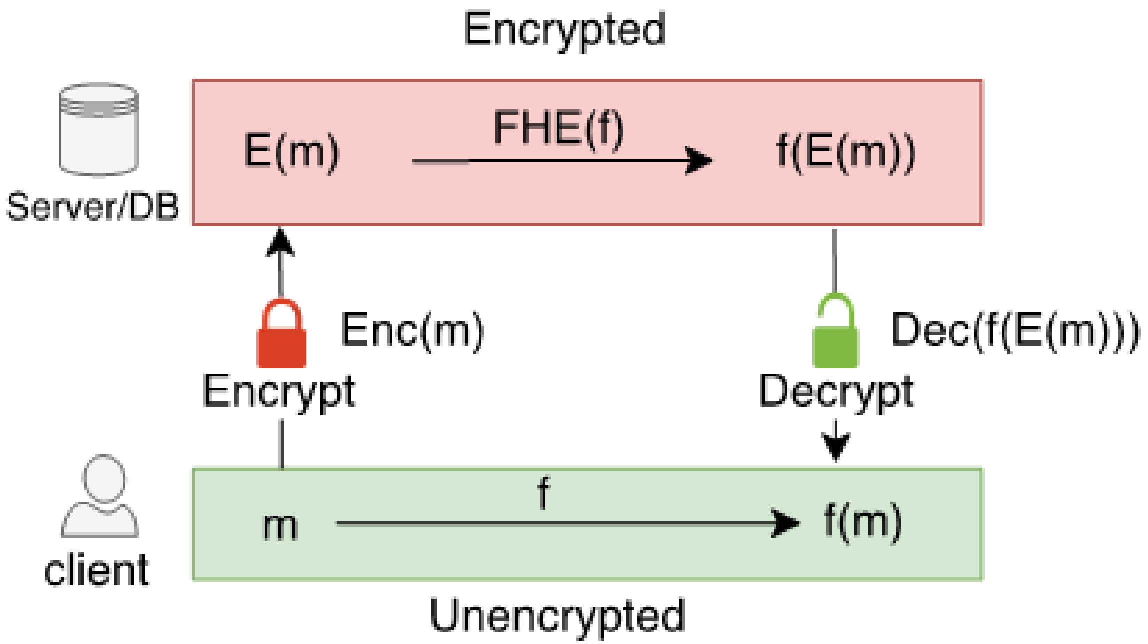
Background - The Privacy Threats

- Online Analytical Processing (OLAP) faces privacy threats
- External attacks:
  - 2022 Uber data breach(What Caused the Uber Data Breach in 2022)
  - 2024 AT&T data breach(AT&T Addresses Recent Data Set Released on the Dark Web)
- Internal leaks:
  - 2023 Tesla former employee information leak(Tesla insider breach exposes thousands of employees)
  - 2024 Evolve Bank information leak(Evolve Bank says ransomware gang stole personal data on millions of customers)

Background - Deficiencies in existing work

- The existing methods cannot meet the requirements of OLAP, or they have deficiencies in terms of efficiency or security.
  - Traditional encryption methods (e.g., AES, RSA) do not support computation.
  - Solutions based on CryptDB support limited computations and may leak data access patterns.
  - Solutions based on TEEs rely on specific hardware and are vulnerable to side-channel attacks.
  - The performance of existing methods based on Fully Homomorphic Encryption (FHE) is not good.
- Among the mentioned methods, we consider the approach based on FHE to achieve the highest level of security. Therefore, we are attempting to enhance the performance of this method.

The Fully Homomorphic Encryption (FHE)



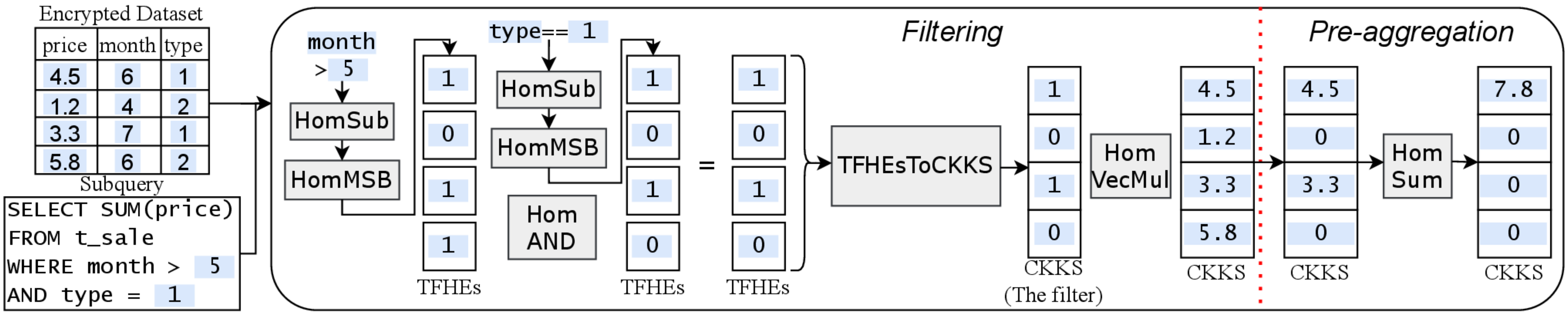
The state-of-the-art single-machine FHE-based OLAP System – HE^3DB

- HE^3DB: An Efficient and Elastic Encrypted Database Via Arithmetic-And-Logic Fully Homomorphic Encryption (https://dl.acm.org/doi/abs/10.1145/3576915.3616608)

- Secure enough
- But with poor performance…

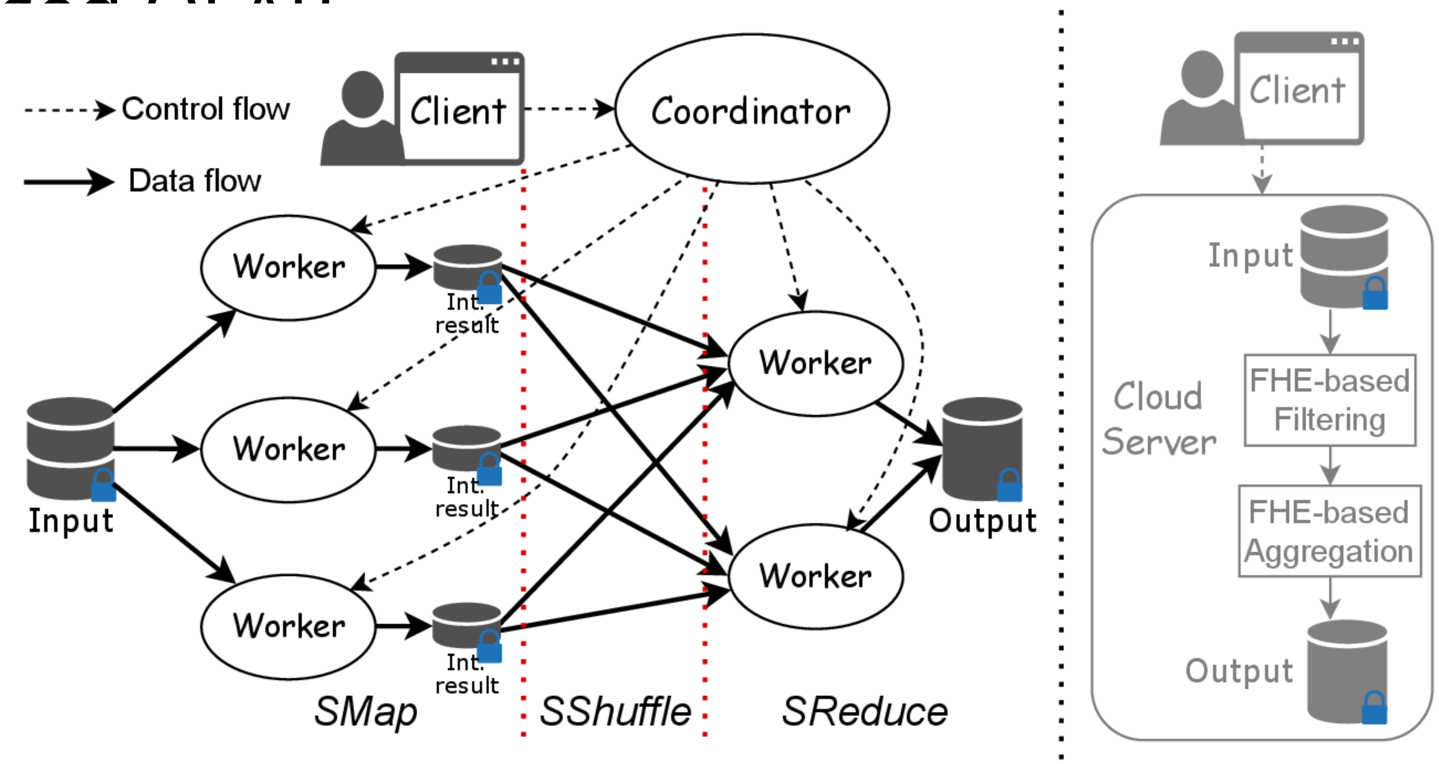
Reasons for Slowness: The FHE-based Filtering

- FHE-based filtering:
  - Utilizing TFHE for homomorphic subtraction on all data entries
  - Performing homomorphic most significant bit (MSB) extraction
  - Conducting homomorphic bitwise AND operations
  - Transforming the results into CKKS format to generate an encrypted 0/1 vector
  - Performing homomorphic vector multiplication between the filter and the data
- Performance bottleneck: 99.96% of the overall time



Key idea

- Can we try using parallel or distributed methods to enhance performance?
- The first step: Employing the MapReduce paradigm for distributed FHE-based OLAP



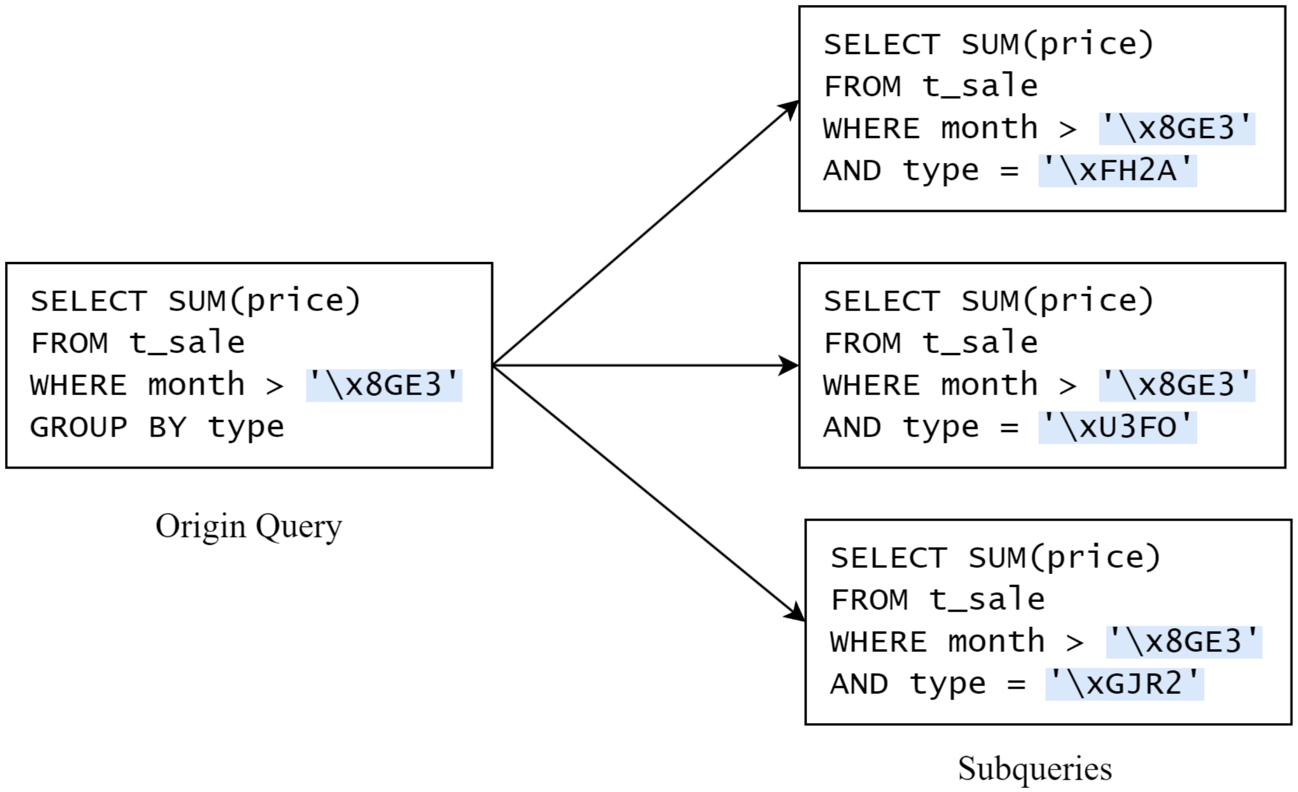
Challenge

- The result of FHE comparison is encrypted randomly, making it impossible to group the intermediate results
- Cannot directly use MapReduce…

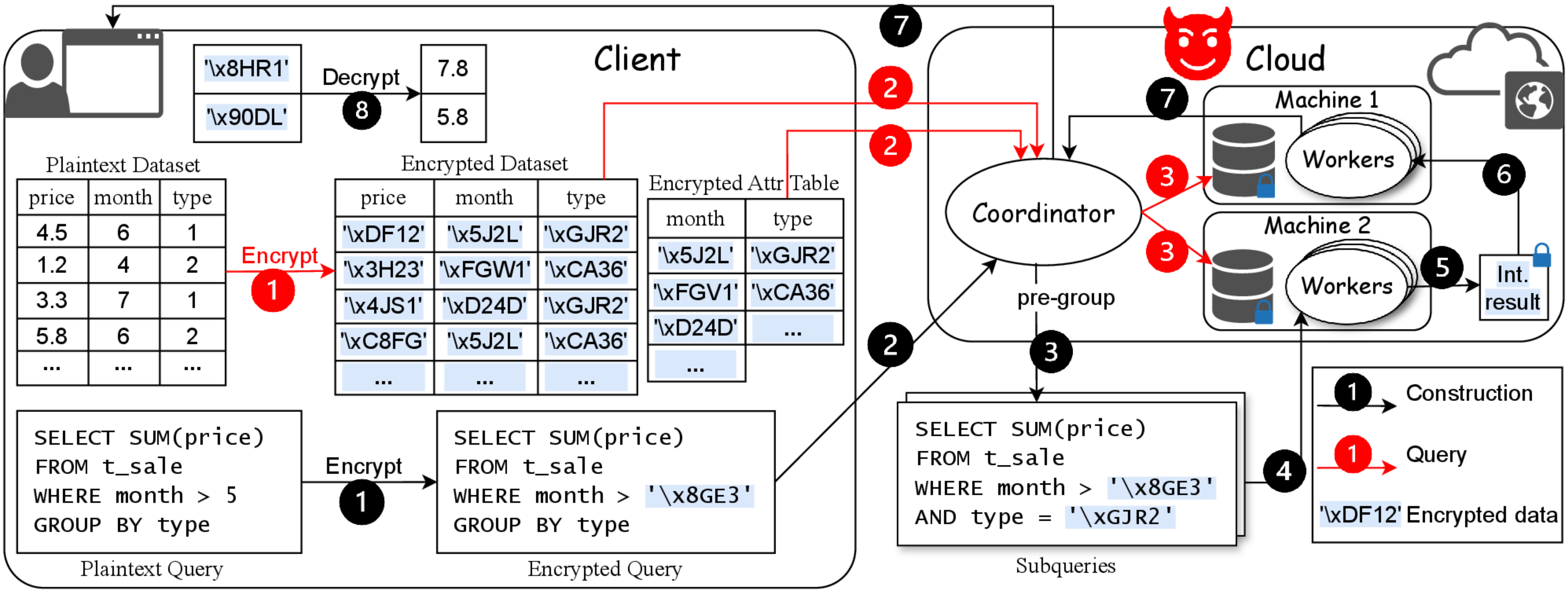
- Can we modify the MapReduce process based on the characteristics of FHE while ensuring security?

The Pre-Group Operation

- Essentially, it involves partitioning queries in advance using attribute tables, breaking down a single query into multiple subqueries.



The Overview of HEDAS



System Construction

Algorithm 1: System Construction

Input: Original plaintext dataset D_{pl}

// Step ❶: Encrypt dataset and generate attributes table

$pk \leftarrow$ `GetEncryptionKey()`; $ek \leftarrow$ `GenerateEvalKey(`pk`)`;

$D_{en} \leftarrow$ `Encrypt(`D_{pl},pk`)`; $attr\_table \leftarrow$ `GenerateAttributeTable(`D_{pl},pk`)`;

// Step ❷: Send encrypted data and attributes table to coordinator

`SendInitData(`D_{en}, $attr\_table$, ek`)`;

// Step ❸: Shard encrypted data and distribute to cloud machines

$machine\_num \leftarrow$ `GetMachineNum()`;

$shards\_array[] \leftarrow$ `DataSharding(`D_{en},$machine\_num$`)`;

For each cloud machine m_i**; do**

  `SendShard(`$shards\_array[m_i]$`)`;

Query Processing

Algorithm 2: Query Processing

Input: Plain text query scheme QS with plaintext filtering predicate parameters P_{pl} and `GROUP BY` attribute G

// Step ❶: Encrypt query parameters
$pk \leftarrow$ `GetEncryptionKey()`; $P_{en} \leftarrow$ `Encrypt(`P_{pl}`,`pk`)`;
// Step ❷: Send query with encrypted parameters to coordinator
`SendQuery(`QS, P_{en}, G`)`;
// Step ❸: pre-group operation
$attr\_table \leftarrow$ `GetAttributeTable()`;
$sub\_queries[] \leftarrow$ `PreGroup(`QS, P_{en}, G, $attr\_table$`)`;
// Step ❹: Assign SMap tasks to worker nodes
While SMap not finished; do
  $worker \leftarrow$ `WaitWorker()`; $SMap\_task \leftarrow$ `GetSMapTask(`$sub\_queries$`)`;
  `AssignTask(`$worker$, $SMap\_task$`)`;
// Step ❺: Workers process SMap tasks
$SMap\_task \leftarrow$ `RequestSMapTask()`;
$int\_result \leftarrow$ `DoSMapTask(`$SMap\_task$`)`; `CacheIntResult(`int_result`)`;
// Step ❻: Assign SReduce tasks
While SReduce not finished; do
  $worker \leftarrow$ `WaitWorker()`; $SReduce\_task \leftarrow$ `GetSReduceTask()`;
  `AssignTask(`$worker$, $SReduce\_task$`)`;
// Step ❼: Workers process SReduce tasks
$SReduce\_task \leftarrow$ `RequestSReduceTask()`;
$int\_results \leftarrow$ `GetIntResults(`$SReduce\_task.key$`)`;
$partial\_result \leftarrow$ `DoSReduceTask(`$SReduce\_task$, $int\_results$`)`;
`SendResultToCoordinator(`$partial\_result$`)`;
// Step ❽: Coordinator collects and returns the final result
$result \leftarrow$ `CollectAllResults()`; `ReturnResult(`$result$`)`;

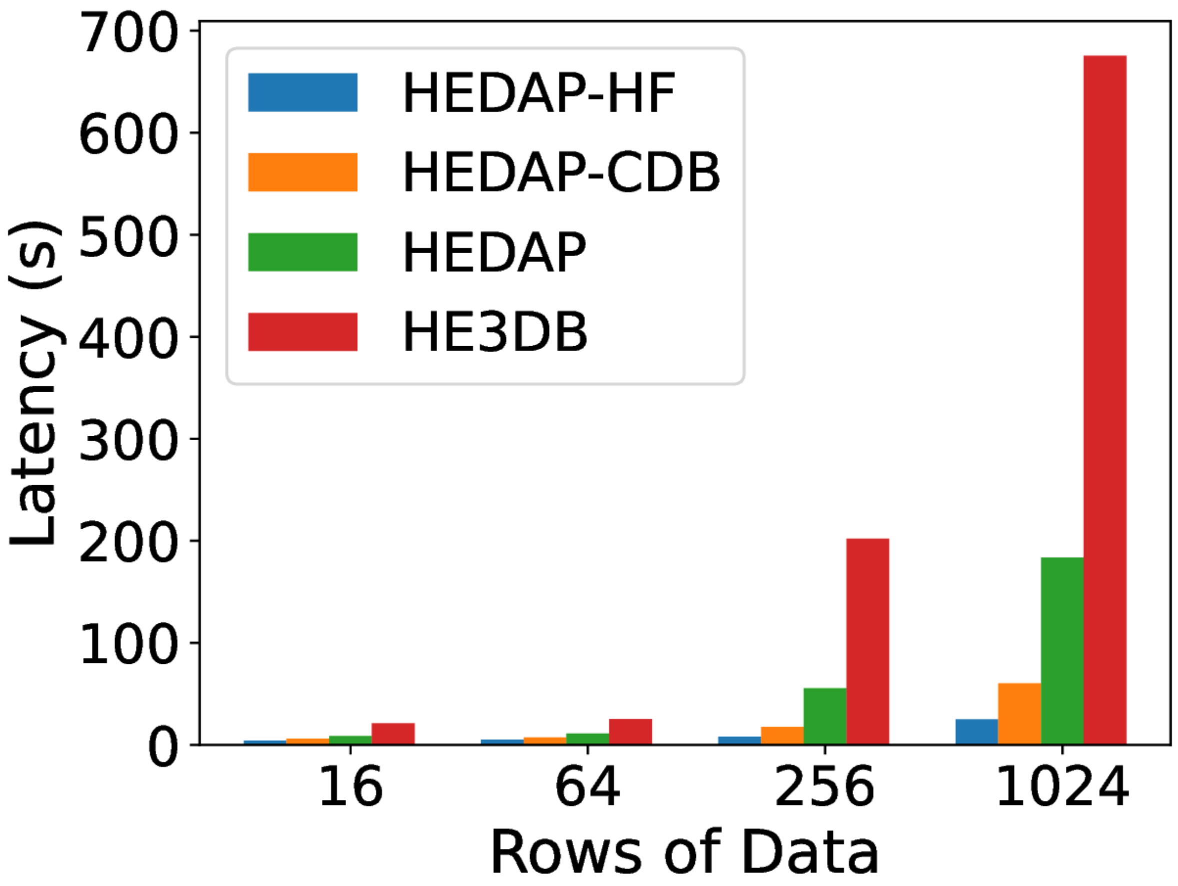
The case studies

- Our system can integrate traditional CryptDB indexes or a type of tree-like HashFilter-based indexes (another work of ours) in scenarios where privacy requirements are not as stringent.
- Two case study subsystems: HEDAS-CryptDB and HEDAS-HashFilter.

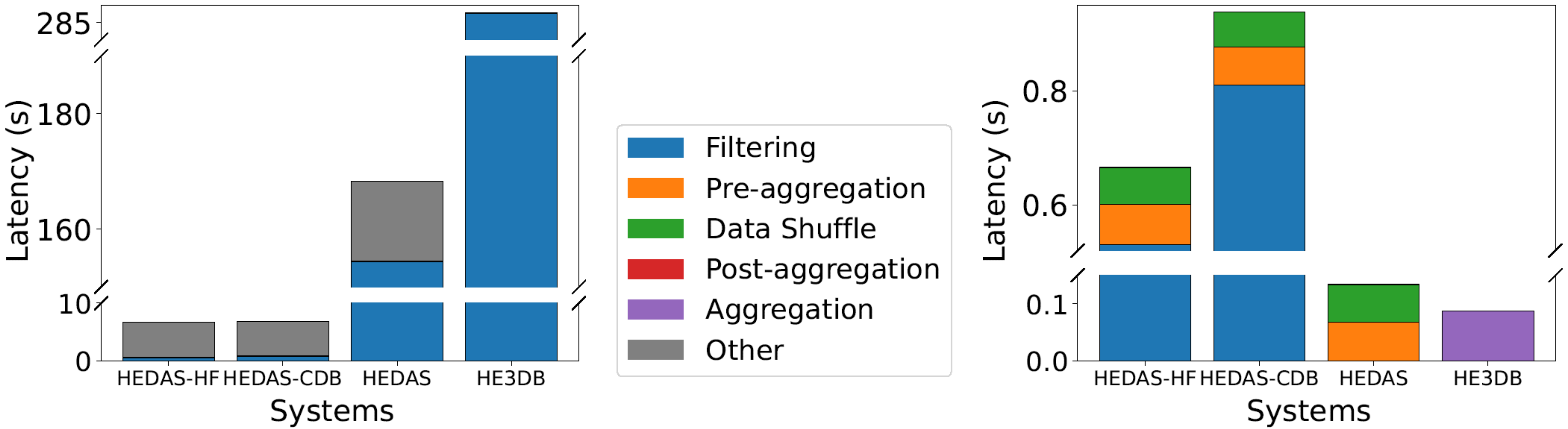
Evaluation questions

- What is the end-to-end and breakdown performance gain compared to HE3DB?
- Additional performance improvement for the case study systems?
- How about the scalability?

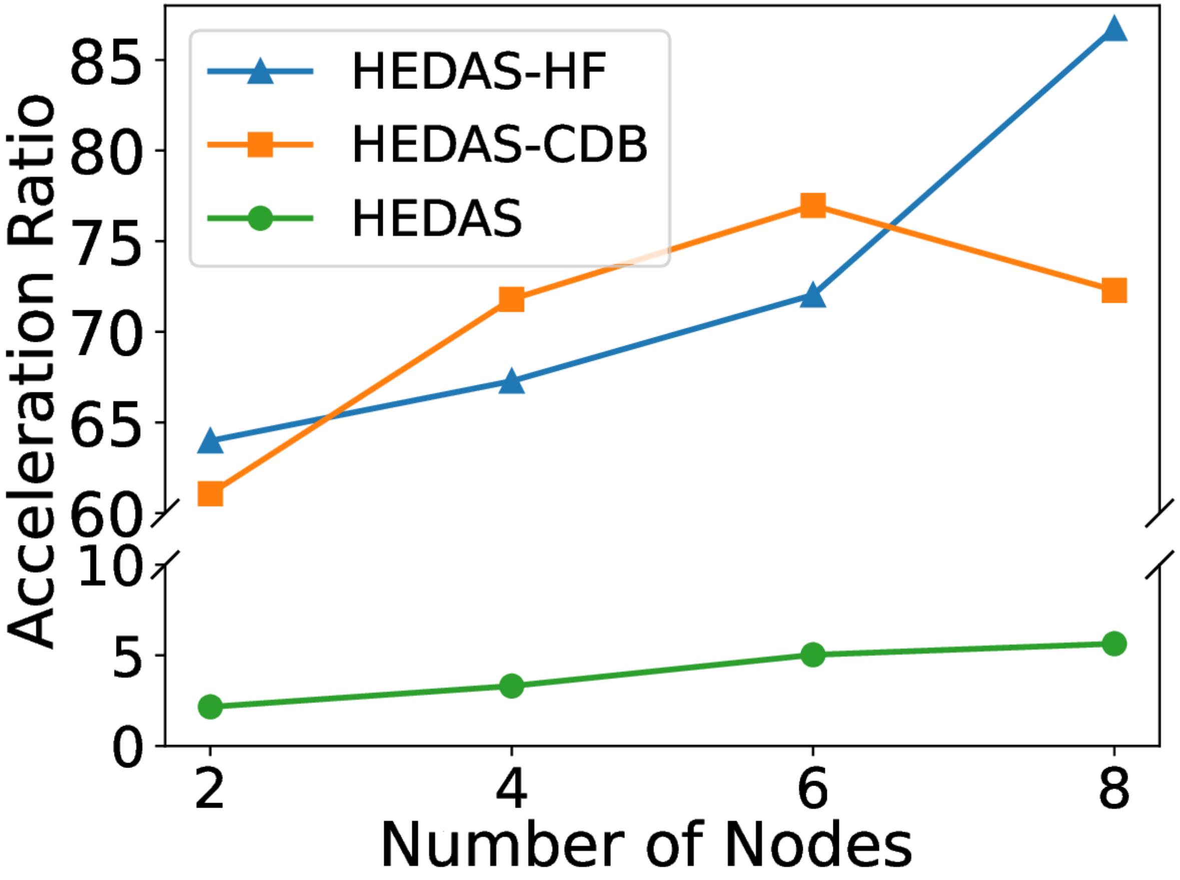
The End-to-end Latency



The Breakdown latency



The Scalability with Increasing Nodes



Conclusion & Future work

- HEDAS is the first distributed FHE-based OLAP system in the MapReduce style, enabling secure and efficient OLAP operations.
- We will continue to explore systematic methods to optimize FHE-based databases, and HEDAS is just the beginning.
- Our future work includes:
  - Utilizing accelerators (e.g., GPUs) to provide more powerful parallel computing;
  - Designing special cache mechanisms to accelerate FHE filtering;
  - Developing more efficient indexing structures for FHE-based databases.

THANK YOU!