CWI

Shielding Data, Embracing Openness, Optimizing Performance: A Journey Through Trustworthy Environments for Database Systems

FOSDEM 2024 - Confidential Computing

Ilaria Battiston, Lotte Felius

February 4, 2024

Who We Are

- PhD students @ CWI Amsterdam
- Our research focuses on *secure databases*
- Encrypted query processing, secure multi-party computation, data privacy...

Our Motivation

CWI

- Protecting the data from intruders while *it is being processed*

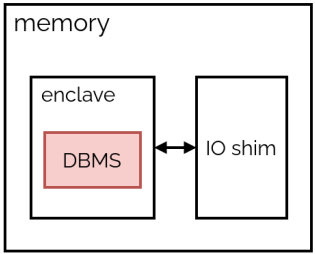
Our Motivation

CWI

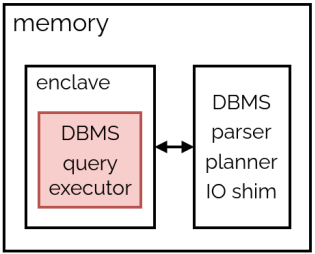
- It is common practice to outsource data storage and management to cloud providers
- Information should be safeguarded from *internal* attacks
  - Homomorphic encryption is still inefficient

→ We employ TEEs as a technology to ensure **confidentiality** and **isolation** of the data

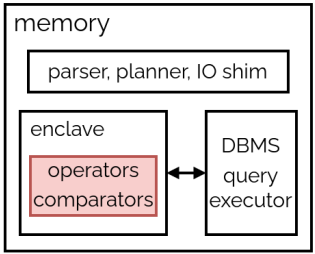
Design Choices[1]



1) full-dbms-split



2) middle-dbms-split

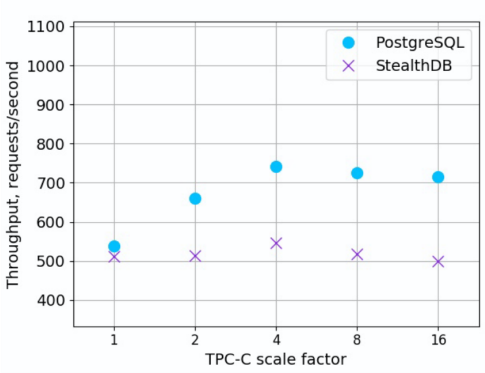


3) minimal-dbms-split

[1]Gribov et al., "StealthDB: a Scalable Encrypted Database with Full SQL Query Support"

StealthDB

- PostgreSQL extension, minimal-dbms-split[2,3]
- 5% to 30% overhead from TPC-C scale factors 1 to 16
- This is as good as it gets with *OLTP** on SGX 1



*transactional workloads, `INSERT`/`UPDATE`-heavy, current data

[2] Gribov et al., "StealthDB: a Scalable Encrypted Database with Full SQL Query Support"

[3] *https://github.com/cryptograph/stealthdb*

Other Databases on SGX 1

- ***SQLite*** → various independent implementations, entirely in enclaves[4]
- ***MariaDB*** → *EdgelessDB*[5], encrypted storage with TLS connections
- ***MS SQL Server*** → *EnclaveDB*, in-memory engine with integrity checks, limited enclave size
- *ObliDB*[6], oblivious physical operators for *OLAP** in the cloud

*analytical workloads, showing insights on historical data

[4] *https://github.com/yerzhan7/SGX_SQLite, https://github.com/wangsu502/SGX-SQLite3*

[5] *https://github.com/edgelesssys/edgelessdb - was at FOSDEM!*

[6] *https://github.com/SabaEskandarian/ObliDB*

Our Contribution

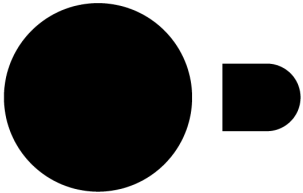
CWI

- DBMS on TEEs primarily target transactional workloads
- There is a lack of research on SGX 2 and its capabilities

♥ We aim to bridge the gap between **efficient** and **secure** *analytical* processing

DuckDB

- Open-source embedded columnar analytical system, born at CWI[7]
- Written in C++ 11, without additional dependencies[8]
- Ported to SGX 1 in 2021[9]



[7] Raasveldt and Mühleisen, "DuckDB: an Embeddable Analytical Database"

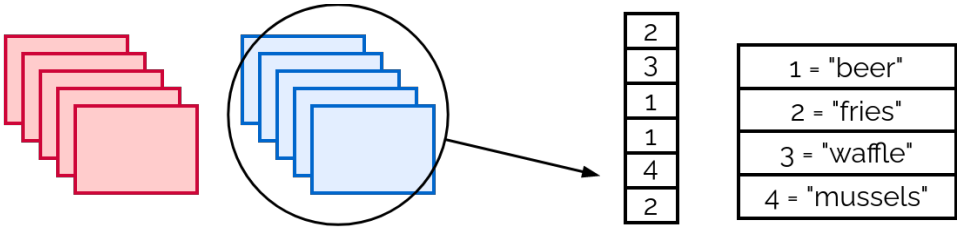
[8] *https://github.com/duckdb/duckdb*

[9] Ansmink, "Encrypted Query Processing in DuckDB"

Columnar Compression

- *Compressed execution*: data is kept compressed until needed
- Decompressed data can be larger than memory

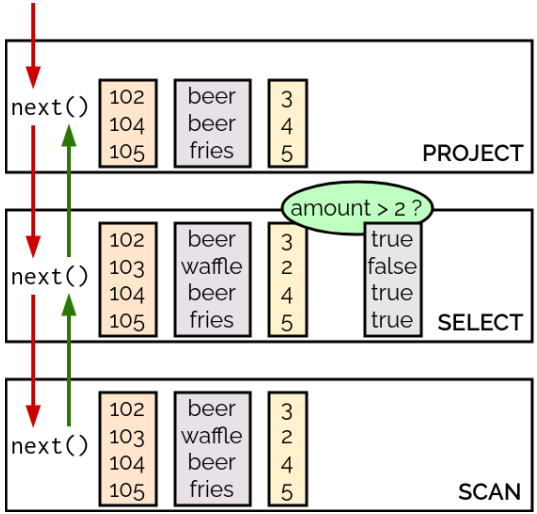
→ we can easily put 10GB of data into an 8GB enclave



Vectorized Execution

```
SELECT id, name, amount
FROM food
WHERE amount > 2;
```

- `next()` returns many tuples, rather than one at the time



DuckDB on SGX 1

- Full-dbms-split
  - Main issue: syscalls are not directly callable
  - 22x slowdown of naive DuckDB on Graphene[10] due to EPC swapping (TPC-H SF1)

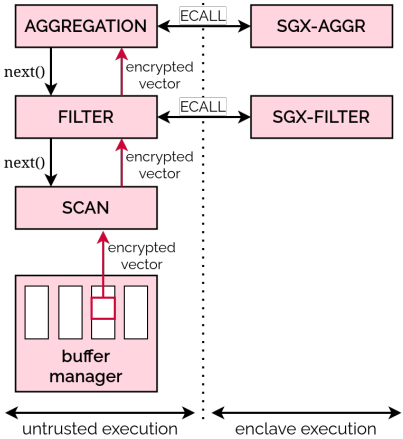
→ Graphene-aware DuckDB

  - Significant speedup, but still 13x slowdown

[10] *https://github.com/gramineproject/gramine*

DuckDB on SGX 1

- Minimal-dbms-split
  - *Vectorized processing* inside the enclave
  - ECalls and OCalls are replaced by asynchronous requests in a shared buffer
  - 10x slowdown (TPC-H SF1, Q6)



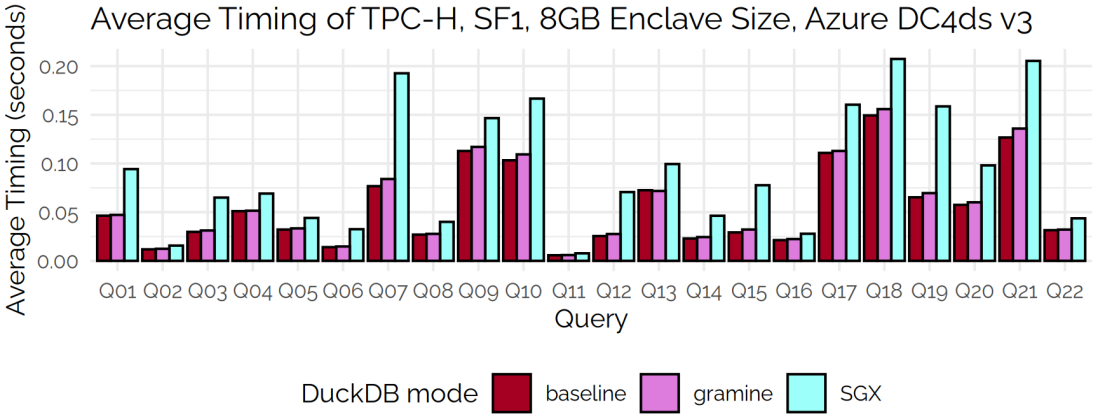
DuckDB on SGX 2

CWI

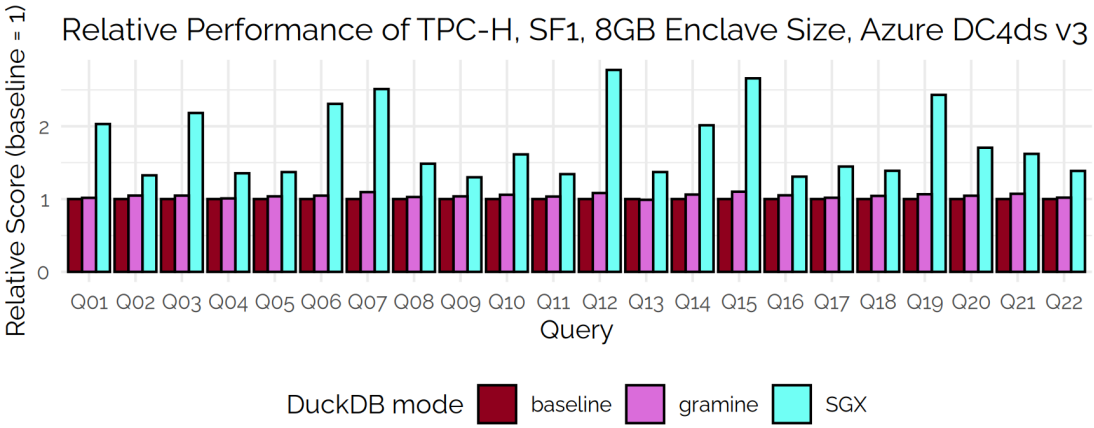
- We leverage the increased EPC capacity to run *natively* DuckDB on Gramine, full-dbms-split[11]

[11] *https://github.com/cwida/DuckDB-SGX*

Preliminary Results



Preliminary Results



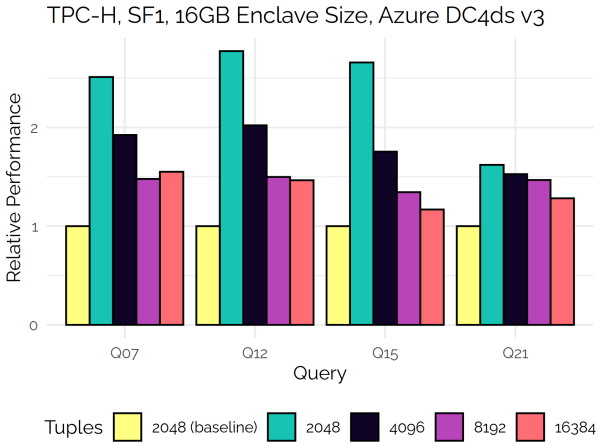
Performance Overhead

CWI

- Mainly caused by `EENTER`/`EEXIT` (OCALLS)
- 2x page faults
- Mitigated by increasing DuckDB vector size

Impact of Vector Size

- Default is 2048 tuples
- Low L1 cache misses, but can incur many EPC calls
- Increasing the vector size improves performance



Conclusions & Future Research

CWI

- Analytics on SGX 2 can be performed efficiently
- We can protect data in secure *memory* – but what about data in *unsecure* memory?
- Goal is to build a **functional** and **fully secure** design for DuckDB

- Stay tuned 😎

Backup Slide - Azure Machine

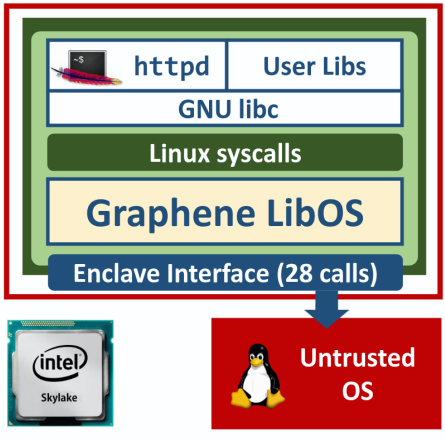
CWI

Azure DC4ds v3:

- 3rd Generation Intel® Xeon Scalable processor
- 4 physical cores
- 32GB memory, 300GiB SSD storage
- 4 max NICs, 16GB EPC memory

Backup Slide - Gramine

- Guest OS designed to run a Linux application with minimal host requirements[12]
- Intercepts all application requests to the host OS
- Allows to run unmodified applications on SGX



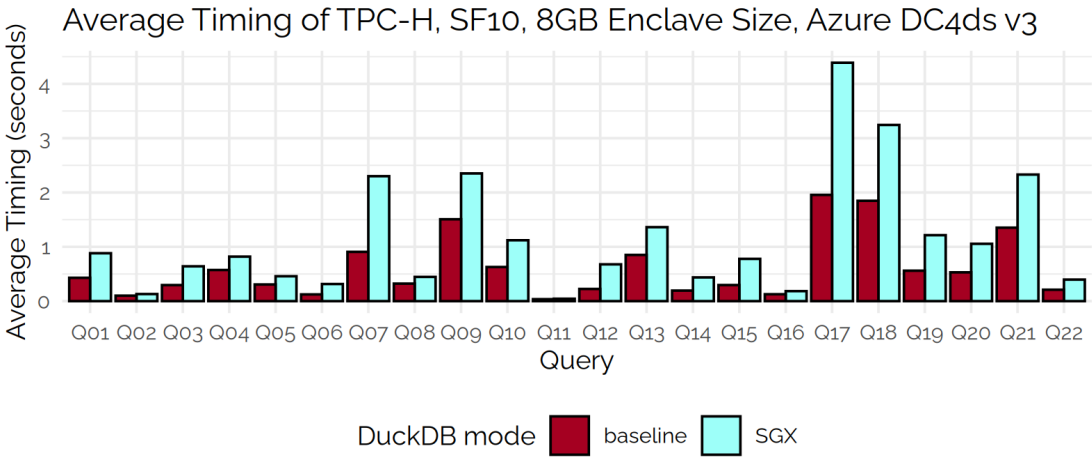
[12] Tsai, Porter, and Vij, "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX"

Backup Slide - TPC-H

CWI

- Widely used benchmark in the field of analytical database systems
- Consists of a set of 22 standardized analytical queries
- Data warehousing and business intelligence applications
- Performance is measured in terms of throughput and response time

Backup Slide - SF10 (Average)



Backup Slide - SF10 (Relative)