Technische Universität Braunschweig

Hochschule Harz

# Scalable Hardware-Aided Trusted Data Management (STAN)


Nico Weichbrodt, 2017-08-28

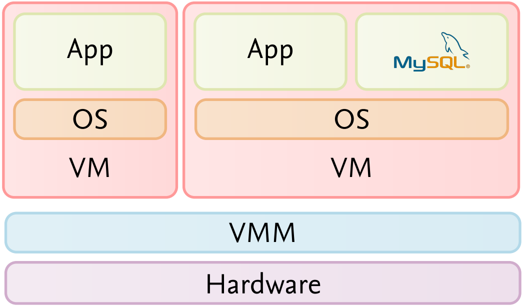
Technische Universität Braunschweig, Hochschule Harz

# Motivation

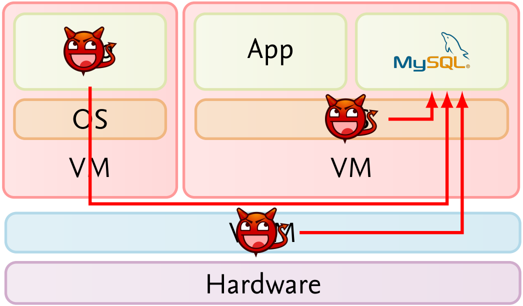
- DBMS' are widely used to store (and sometimes process) data
- Either on-premise or at a cloud provider somewhere

- Do you trust those providers?
- Would you let them store your sensitive data?

# Attack Scenarios



# Attack Scenarios




- Adversaries: other customers, the provider itself, provider staff, ...
- Bugs in OS, VMM, Apps...

# Previous Approaches

- **Proxy-based security** (Mylar)
  - Move query processing to the client, server only as storage
  - ✗ Data confidentiality but moves processing to clients

---

[1] Trusted Computing Base

[2] Trusted Execution Environment

# Previous Approaches

- **Proxy-based security** (Mylar)
  - Move query processing to the client, server only as storage
  - ✗ Data confidentiality but moves processing to clients
- **Data encryption** (CryptDB)
  - Execute queries over encrypted data (homomorphic encryption)
  - ✗ Slow and restricts query types

[1] Trusted Computing Base
[2] Trusted Execution Environment

# Previous Approaches

- **Proxy-based security** (Mylar)
  - Move query processing to the client, server only as storage
  - ✗ Data confidentiality but moves processing to clients
- **Data encryption** (CryptDB)
  - Execute queries over encrypted data (homomorphic encryption)
  - ✗ Slow and restricts query types
- **Trusted subsystem** (Cipherbase, TrustedDB)
  - Split DBMS into trusted and untrusted part
  - ✗ High TCB[1], duplication of functionality, custom hardware

---

[1] Trusted Computing Base
[2] Trusted Execution Environment

# Previous Approaches

- **Proxy-based security** (Mylar)
  - Move query processing to the client, server only as storage
  - ✗ Data confidentiality but moves processing to clients
- **Data encryption** (CryptDB)
  - Execute queries over encrypted data (homomorphic encryption)
  - ✗ Slow and restricts query types
- **Trusted subsystem** (Cipherbase, TrustedDB)
  - Split DBMS into trusted and untrusted part
  - ✗ High TCB[1], duplication of functionality, custom hardware
- **Trusted execution** (Haven, SCONE)
  - Put the unmodified DBMS into a TEE[2] on top a library OS
  - ✗ Large TCB, unrealistic evaluation results

---

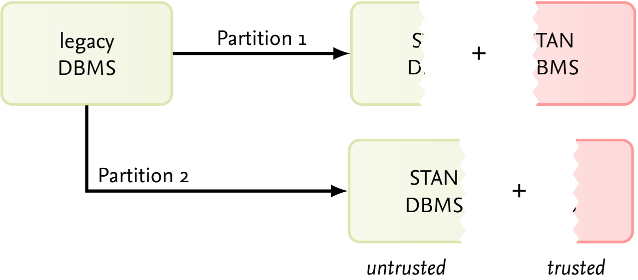
[1] Trusted Computing Base

[2] Trusted Execution Environment

# STAN's Solution

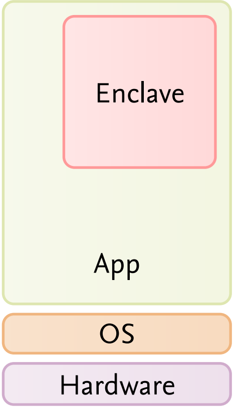
- We know the expected overhead of trusted execution
- Don't put everything into TEE, no library OS
- Different users $\rightarrow$ different requirements

$\rightarrow$ Smartly partition the DBMS using software product line approaches

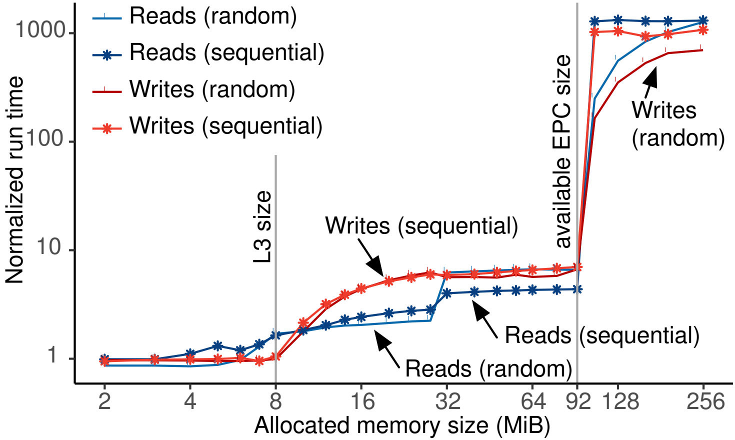


# Intel Software Guard Extensions (SGX)

- Secure compartments called enclaves
- Defined entry points
- Multithreading capable
- Enclave Page Cache (EPC)
  - Memory encryption
  - Confidentiality & integrity protected
  - Limited size (≈93MiB)
  - EPC to memory paging
  - Handled by SGX driver

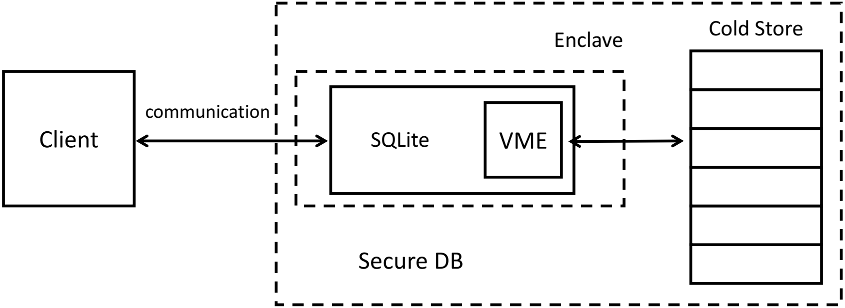


# SGX Worst Case Memory Access Performance

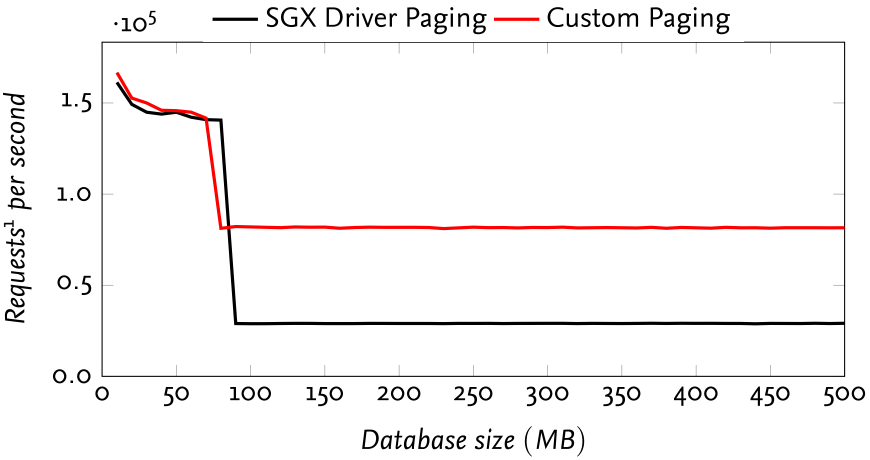



From *SCONE: Secure Linux Containers with Intel SGX*

# SQLite with Custom Paging



# SQLite First Results




[1] 1 Request = 10 SELECTs

# Work Program (excerpt)

- **Proactive Working Set Management** (TUB)
  - Use custom paging algorithms to move data from/to EPC
  - Experiment more with in-EPC compression
- **System Support for Integrity Preservation** (TUB)
  - Integrity protected data processing
  - Detection of roll-back attacks on enclaves

# Work Program (excerpt)

- **Proactive Working Set Management** (TUB)
  - Use custom paging algorithms to move data from/to EPC
  - Experiment more with in-EPC compression
- **System Support for Integrity Preservation** (TUB)
  - Integrity protected data processing
  - Detection of roll-back attacks on enclaves
- **Trust-aware DBMS Architecture** (HSH)
  - Adapt custom paging of DBMS to transparently encrypt data
  - Identify trust dependencies in DBMS components
- **Trusted Query Execution Considering the Users Needs** (HSH)
  - Feature model addressing trusted features
  - Suitable techniques to declare user-defined trusted data regions