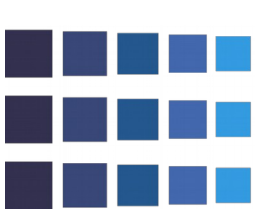


Technion
Israel Institute of Technology

Hasso
Plattner
Institut
IT Systems Engineering | Universität Potsdam

CoSMIX: A Compiler-based System for Secure Memory Instrumentation and Execution in Enclaves

Meni Orenbach
Yan Michalevsky
Christof Fetzer
Mark Silberstein

TECHNISCHE
UNIVERSITÄT
DRESDEN

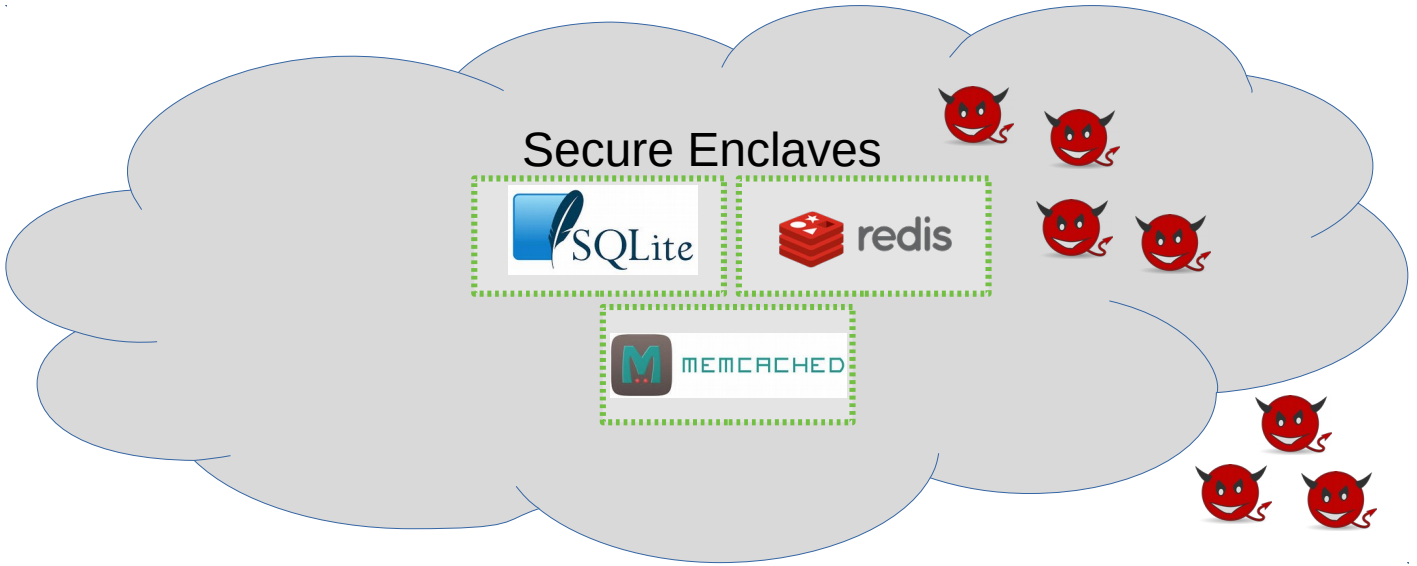
anjuna

The night is dark and full of terrors

The night is dark and full of terrors

The cloud is dark and full of terrors

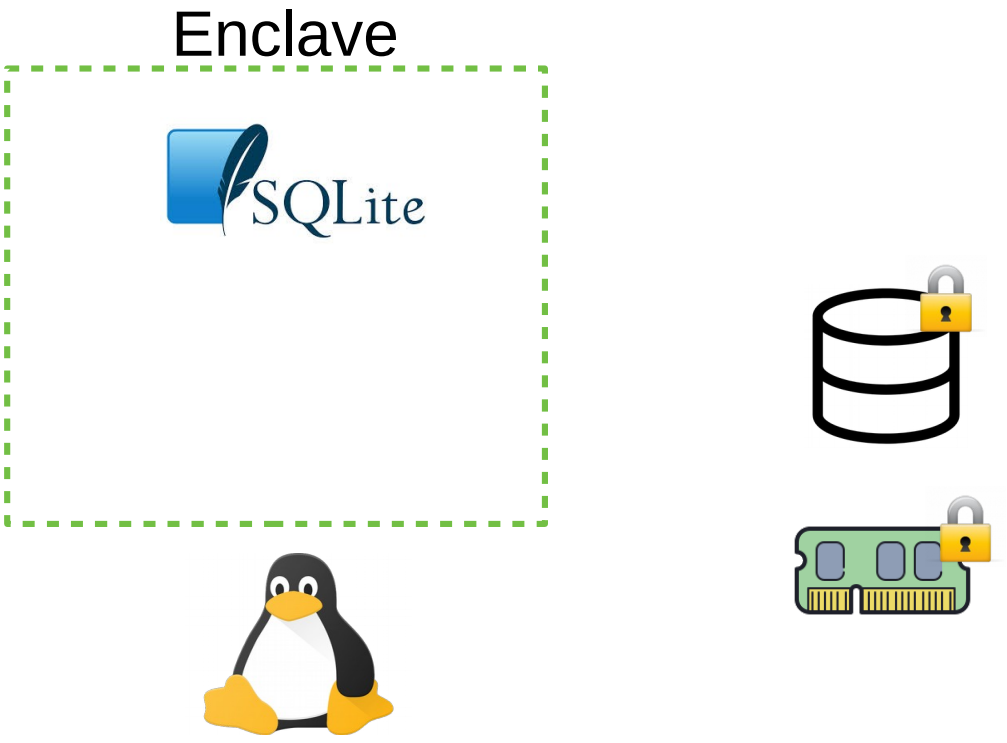
The cloud is dark and full of terrors
Yet, hardware enclaves can help



Enclaves shield applications from privileged adversaries

- Confidentiallity
- Integrity

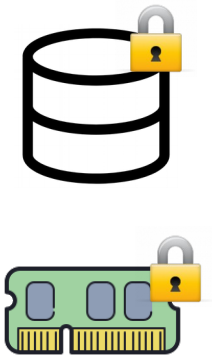
Untrusted OS



Enclaves shield applications from privileged adversaries

- Confidentiallity
- Integrity

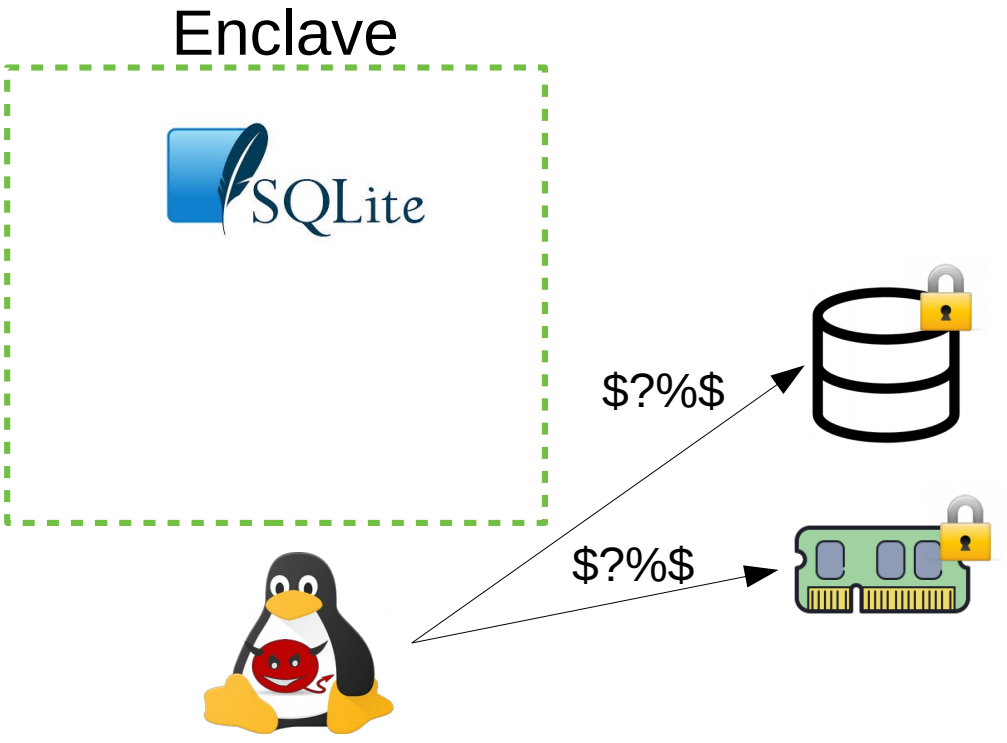
Untrusted OS



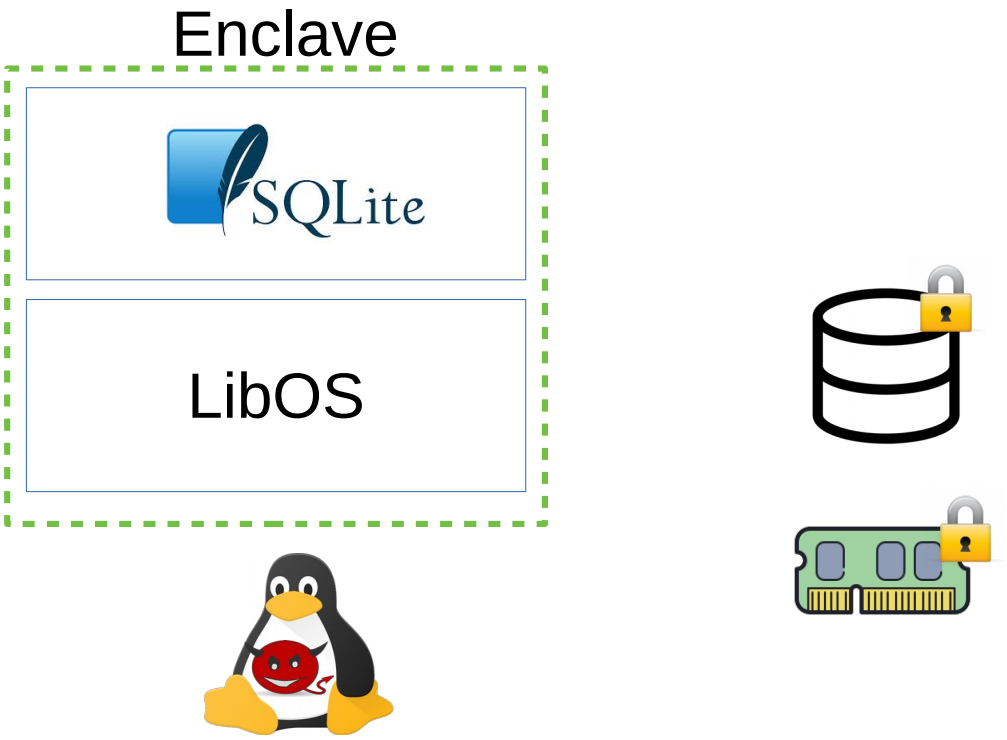
Enclaves shield applications from privileged adversaries

- Confidentiallity
- Integrity

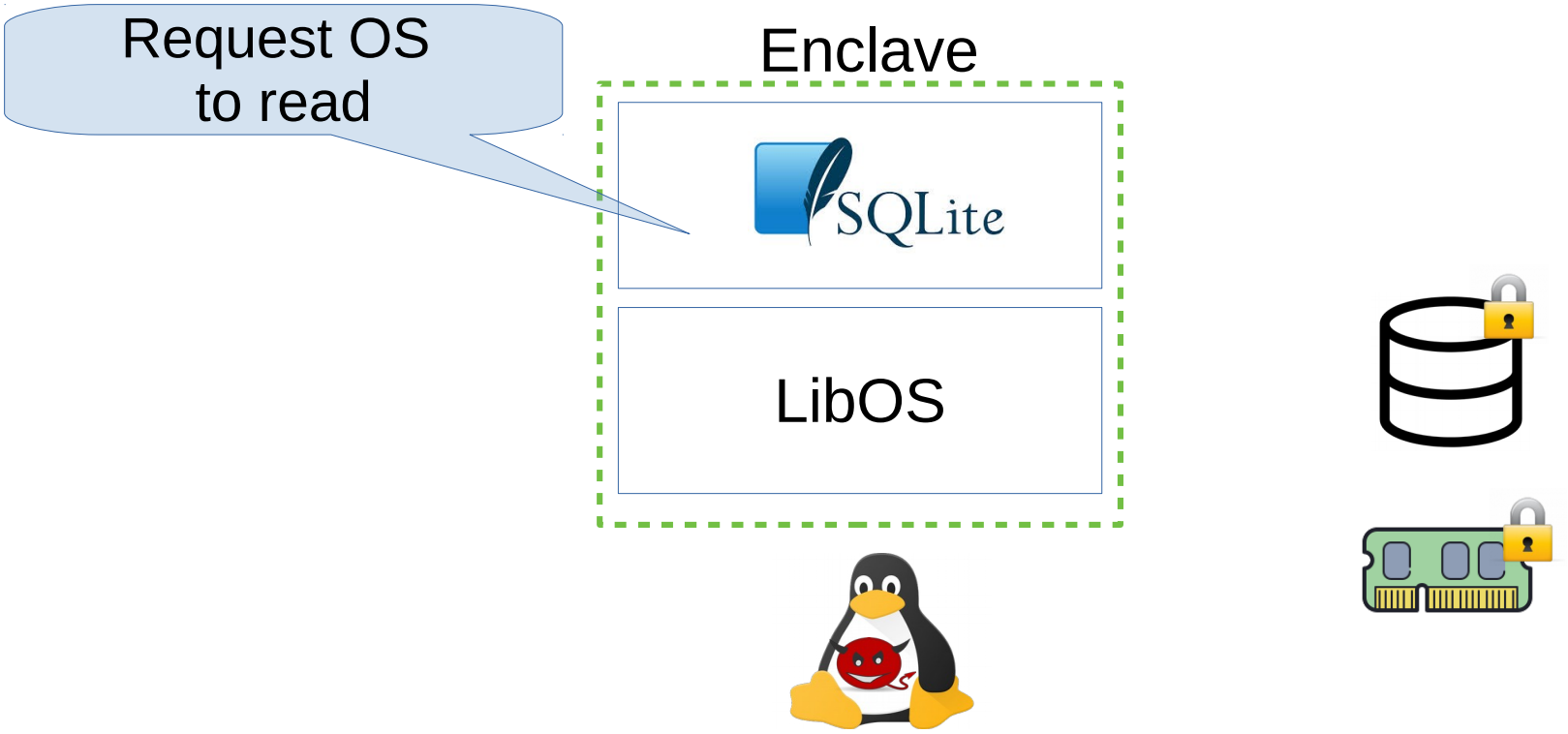
Untrusted OS



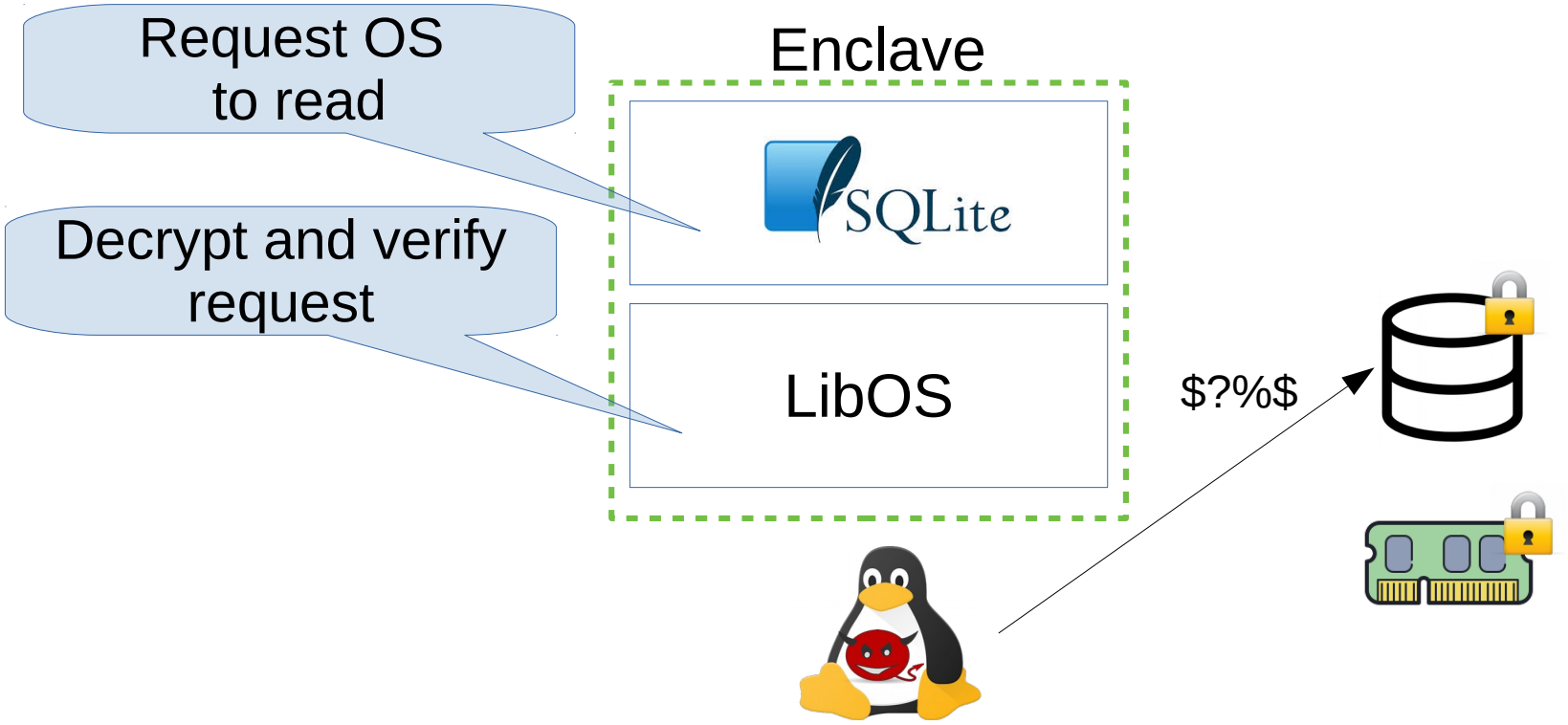
Run unmodified applications inside enclaves



Run unmodified applications inside enclaves



Run unmodified applications inside enclaves



Talk focus: Current SGX Enclaves

Integrated into Intel CPUs

Can we execute

any x86 application

inside **enclaves**?

Talk focus: Current SGX Enclaves

Integrated into Intel CPUs

Can we execute

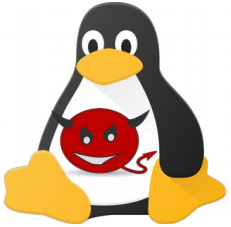
any x86 application

inside **enclaves**?

Memory-mapped files in SGX?

Enclave

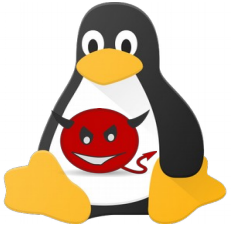

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```



Memory-mapped files in SGX?

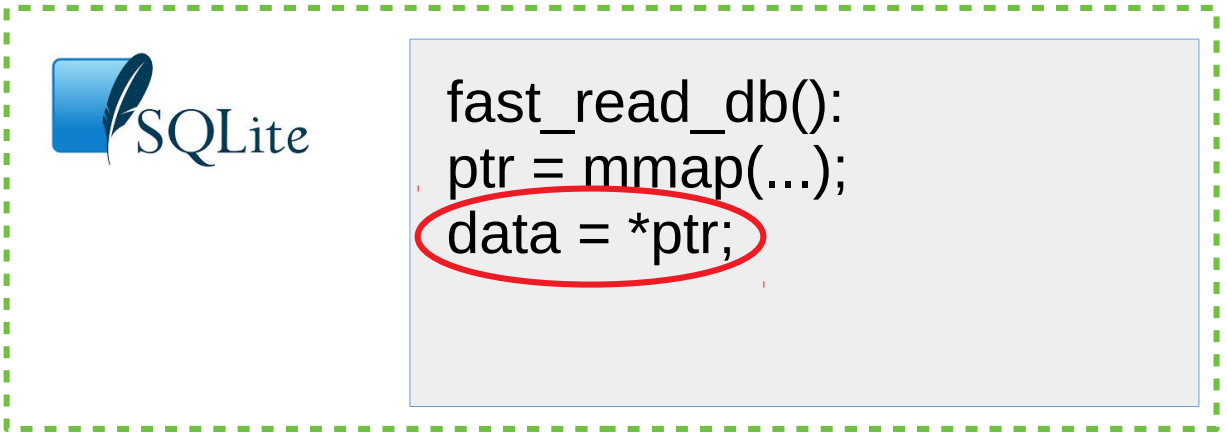
Enclave

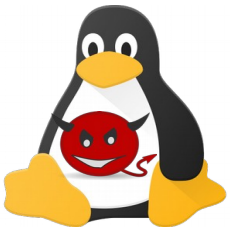

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

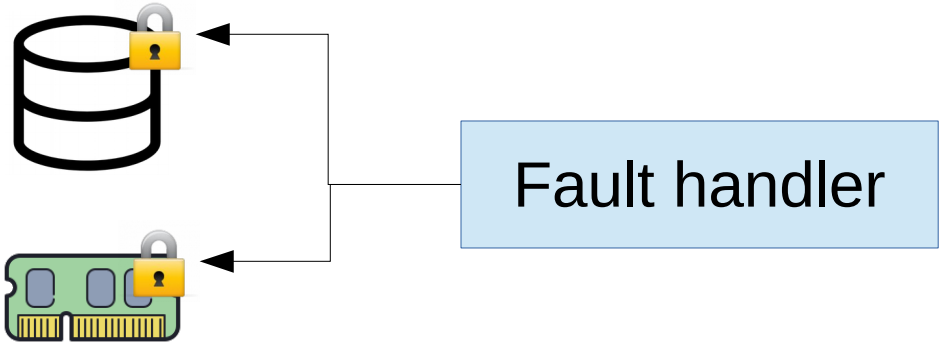


Memory-mapped files in SGX?

Enclave





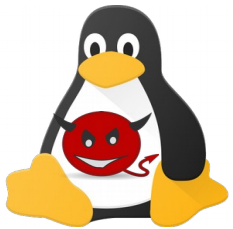


Memory-mapped files in SGX?

Enclave


```
fast_read_db
ptr = mmap(.
data = *ptr;
```

Handler customization is a powerful tool
- File mapping
- Disaggregated memory
- Compressed memory
  and more



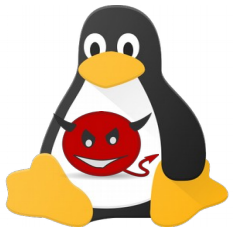
Memory-mapped files in SGX?

Enclave


```
fast_read_db
ptr = mmap(.
data = *ptr;
```

Handler customization is a powerful tool
- File mapping
- Disaggregated memory
- Compressed memory
  and more

Efficient and requires **no modifications**
Applications **rely** on this abstraction



Memory-mapped files in SGX?

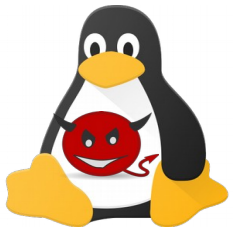
Enclave


```
fast_read_db
ptr = mmap(.
data = *ptr;
```

Handler customization is a powerful tool
- File mapping
- Disaggregated memory
- Compressed memory
  and more

Efficient and requires **no modifications**
Applications **rely** on this abstraction

Its Not Possible in SGX Today!

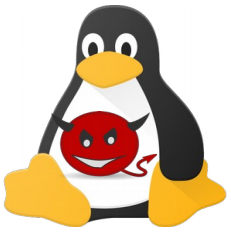


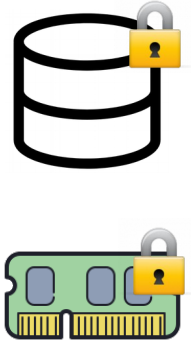
Memory-mapped files in SGX?

Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

Implementation options:
Use fault handler
Use SGX demand-paging instructions
Use in-enclave handler





Fault handler

Memory-mapped files in SGX?

Enclave

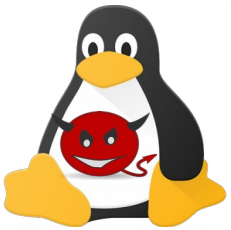

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

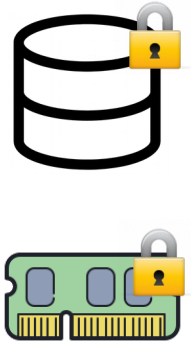
Implementation options:

~~Use fault handler~~

Use SGX demand-paging instructions

Use in-enclave handler





Fault handler

Memory-mapped files in SGX?

Enclave

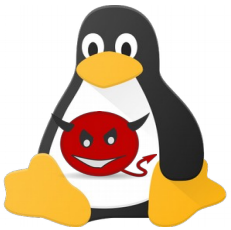

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

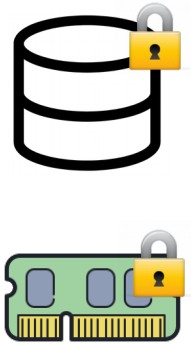
Implementation options:

~~Use fault handler~~

~~Use SGX demand-paging instructions~~

Use in-enclave handler



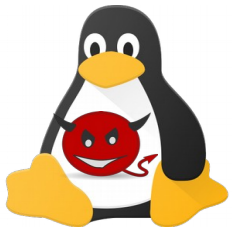


Fault handler

Inefficient: in-enclave handler

Enclave

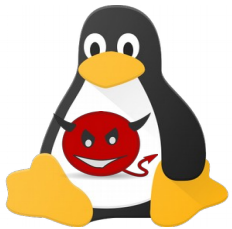

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```



Inefficient: in-enclave handler

Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```



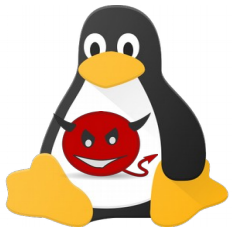
Inefficient: in-enclave handler

Enclave

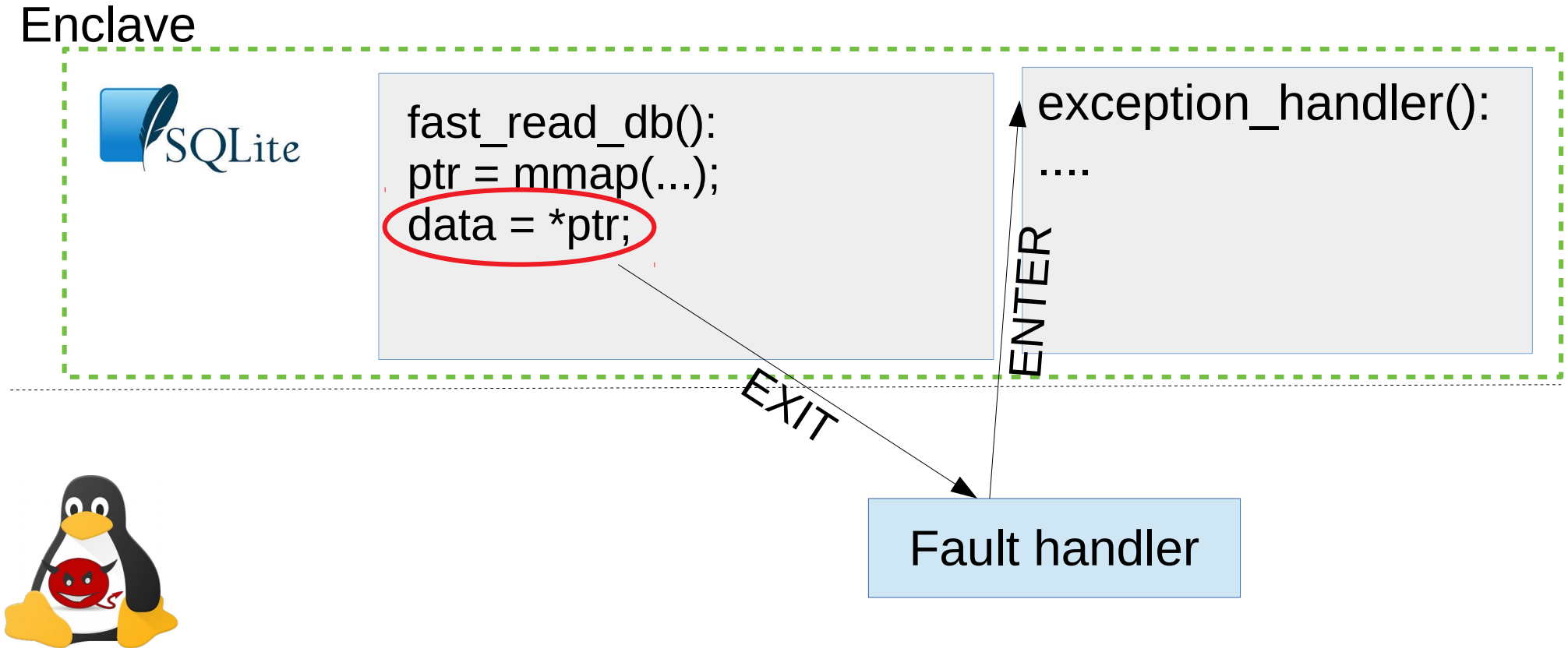

```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

EXIT

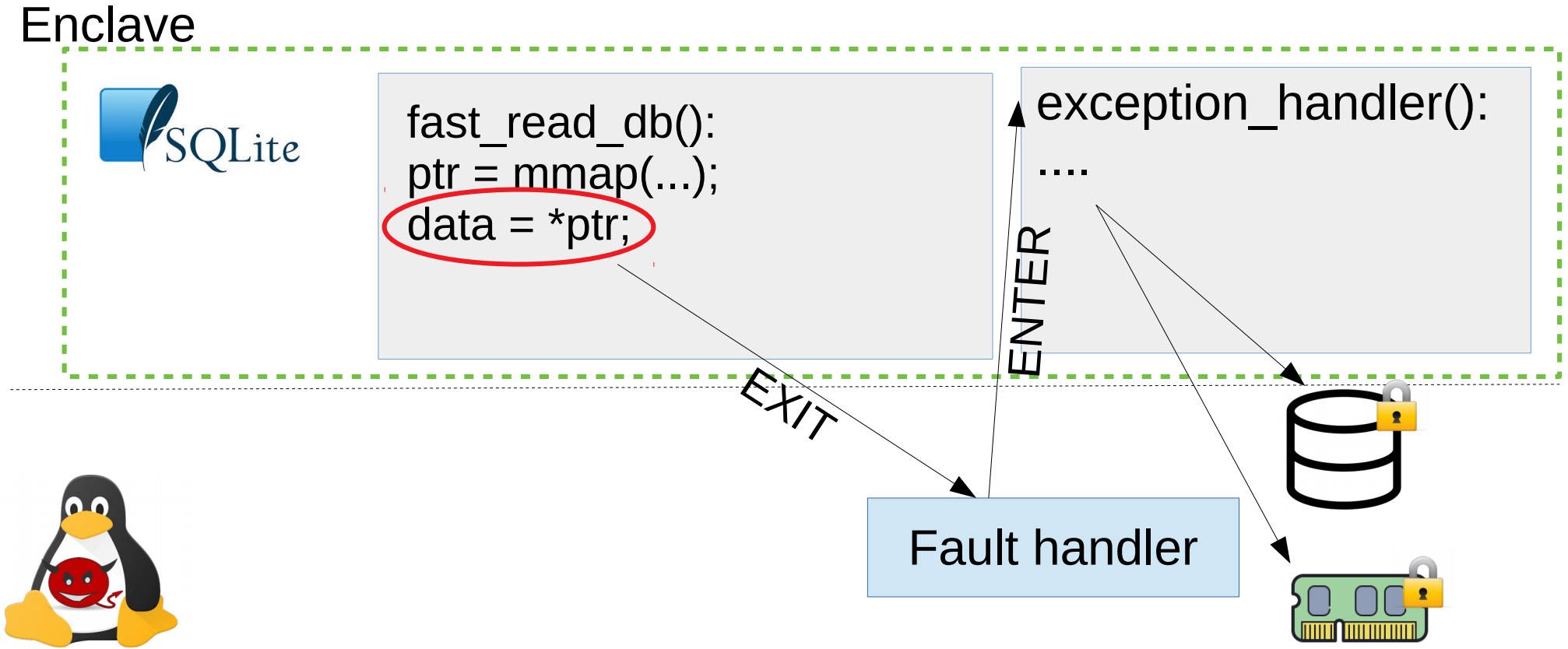
Fault handler



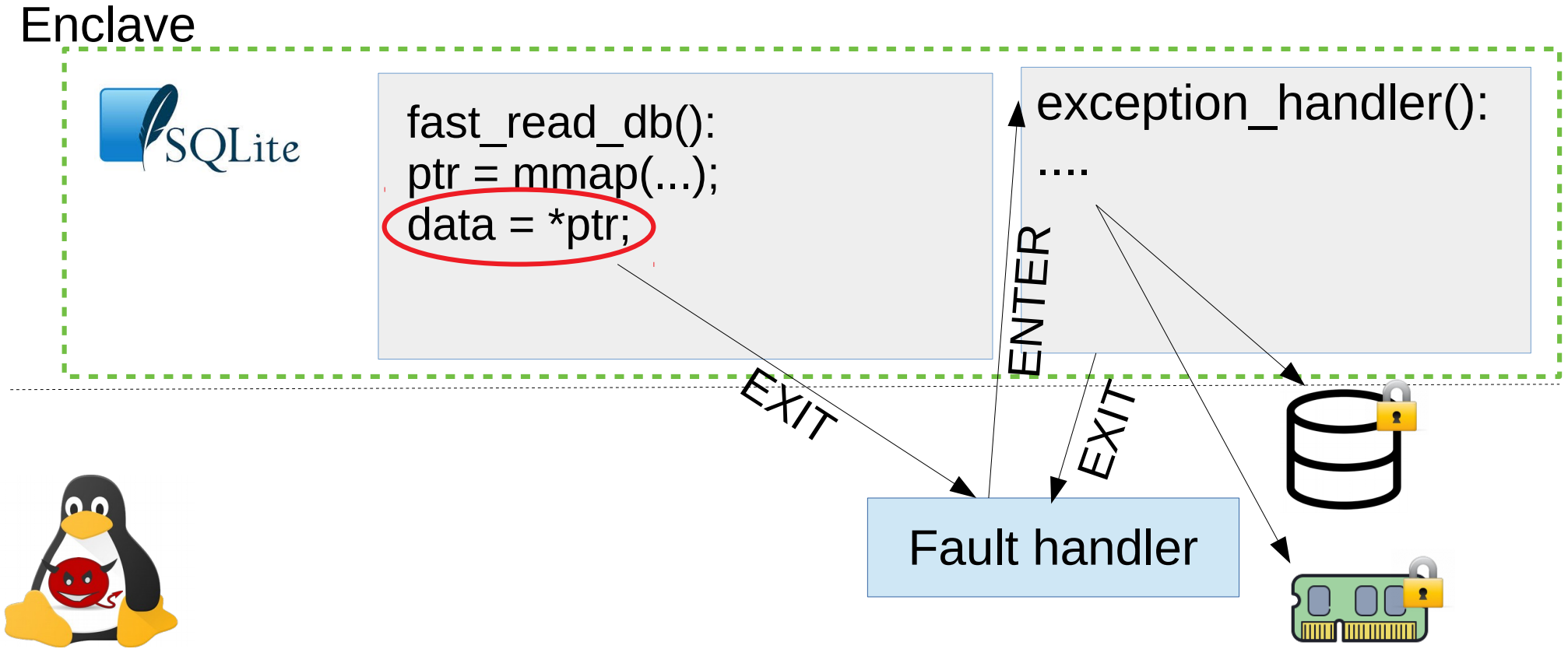
Inefficient: in-enclave handler



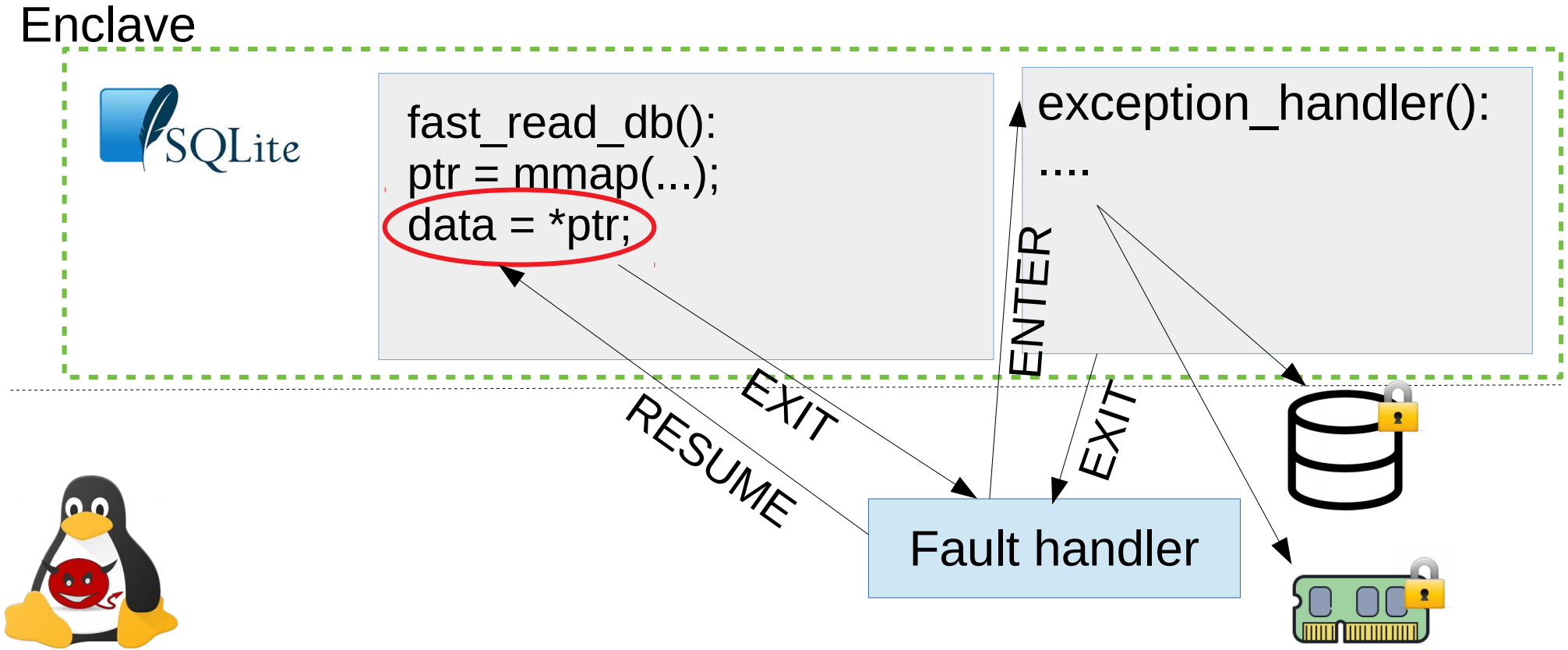
Inefficient: in-enclave handler



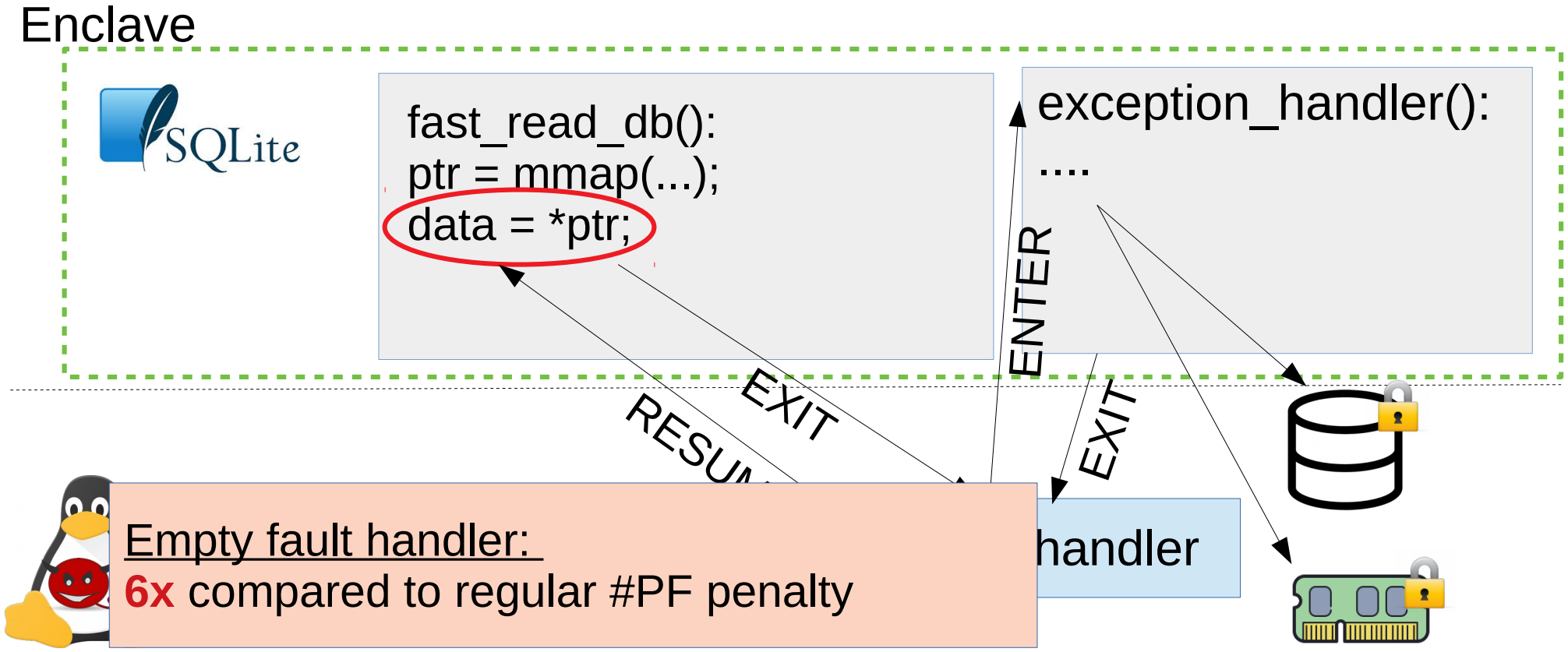
Inefficient: in-enclave handler



Inefficient: in-enclave handler



Inefficient: in-enclave handler

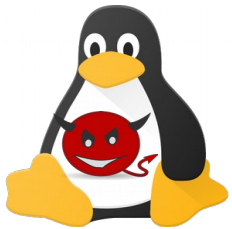


Insecure: in-enclave handler

Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

```
exception_handler():
....
```

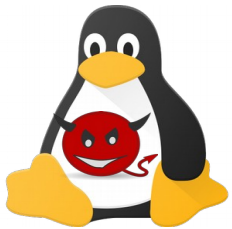


Insecure: in-enclave handler

Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

```
exception_handler():
....
```



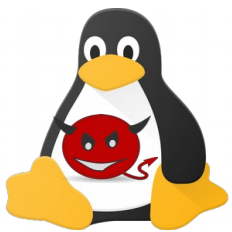
Insecure: in-enclave handler

Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

```
exception_handler():
....
```

EXIT



Insecure: in-enclave handler

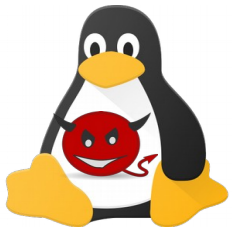
Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

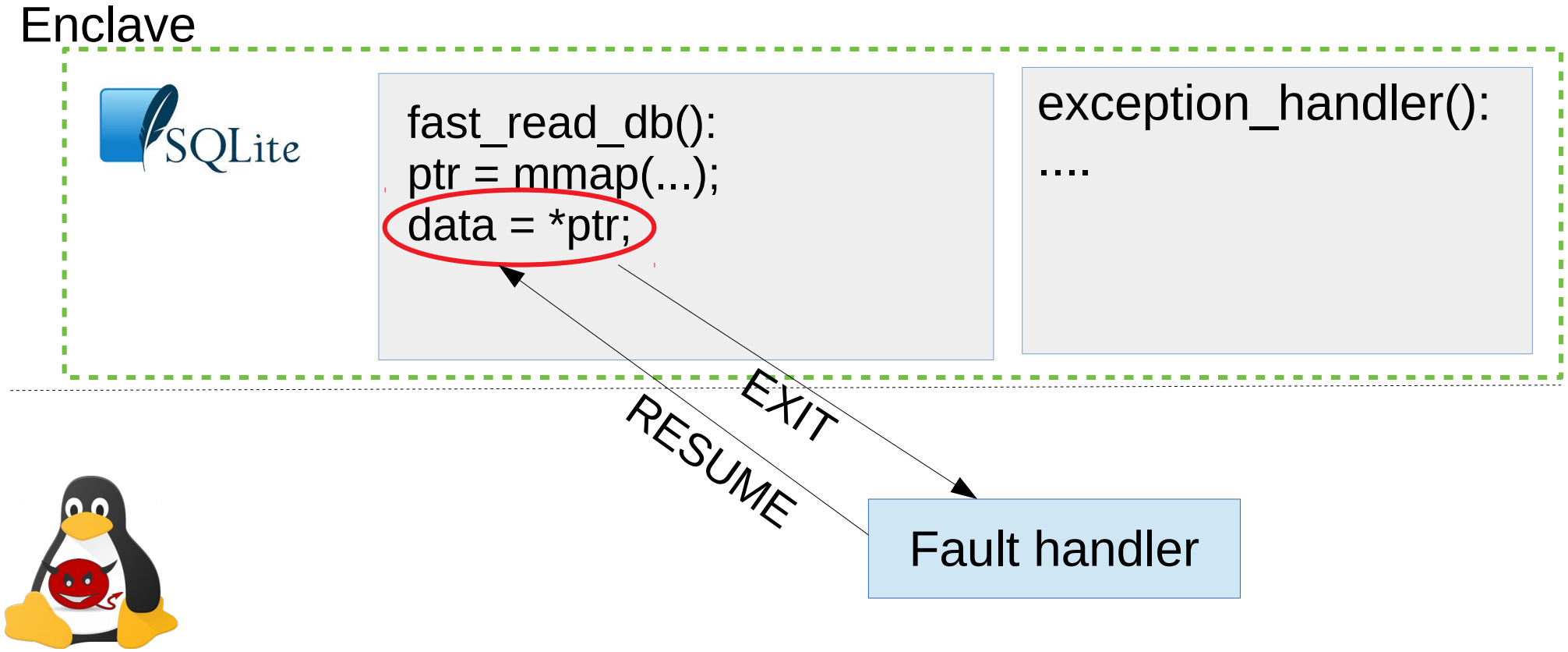
```
exception_handler():
....
```

EXIT

Fault handler



Insecure: in-enclave handler



Insecure: in-enclave handler

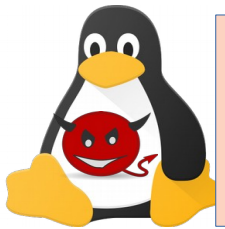
Enclave


```
fast_read_db():
ptr = mmap(...);
data = *ptr;
```

```
exception_handler():
....
```

EXIT

RESUME



Enclaves are missing an OS abstraction!

Fault handler

Previous works that sidestep the lack of secure page faults

- In-enclave paging
- Oblivious page accesses
- Secure access to remote memory

Ad-hoc solutions
Requires applications modifications
What if we want to use more than one?

Previous works that sidestep the lack of secure page faults

- In-enclave paging
- Oblivious page accesses
- Secure access to remote memory

Ad-hoc solutions
Requires applications modifications
What if we want to use more than one?

Efficient page fault customization **missing**
We cannot rely on **hardware**
We cannot **change applications**

So what can we do?

Efficient page fault customization **missing**
We cannot rely on **hardware**
We cannot **change applications**

So what can we do?

Automatically change applicataions with memory instrumentation

Agenda

- ~~Motivation~~
- CoSMIX
- Evaluation

CoSMIX

Efficient **Instrumenting** compiler

and runtime system

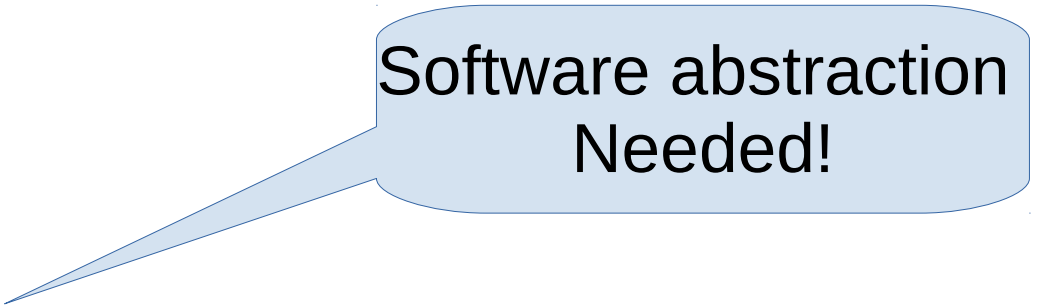
What to instrument?

- Observation:
  - No need to instrument every memory access
- Instrumentation policy:
  - Annotations on memory allocations
  - Automatically infering memory accesses to be instrumented

How to express memory behavior

Interface for custom page fault handlers

How to express memory behavior



Interface for custom page fault handlers

Memory Stores (Mstores)

Another layer of virtual memory

on top of an abstract **backing store**

Memory Stores (Mstores)

Another layer of virtual memory

on top of an abstract **backing store**

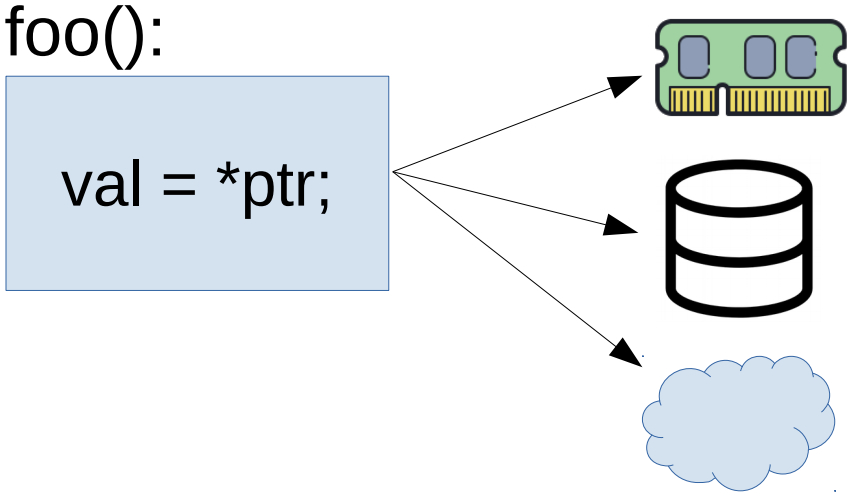
foo():

```
val = *ptr;
```

Memory Stores (Mstores)

Another layer of virtual memory

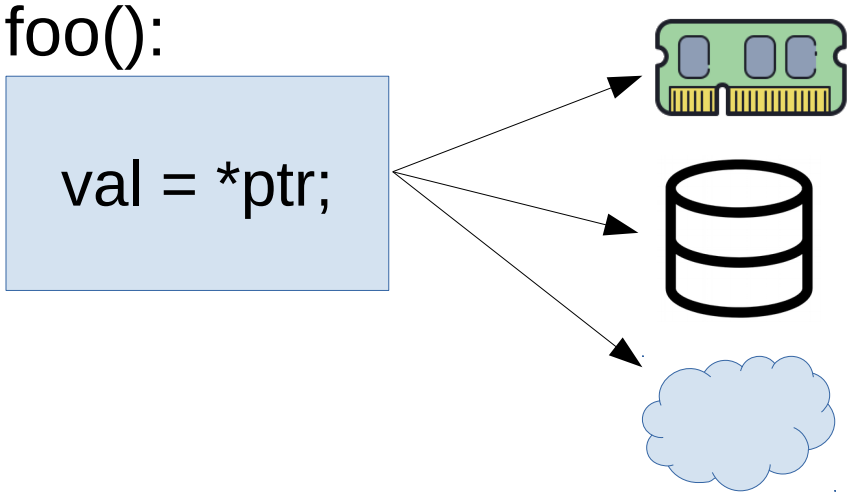
on top of an abstract **backing store**



Memory Stores (Mstores)

Another layer of **virtual memory**

on top of an abstract backing store

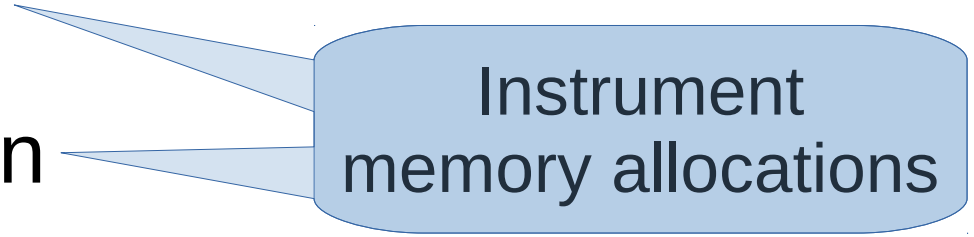


Memory store interface

- Allocation
- Deallocation
- Address translation
- Paging system

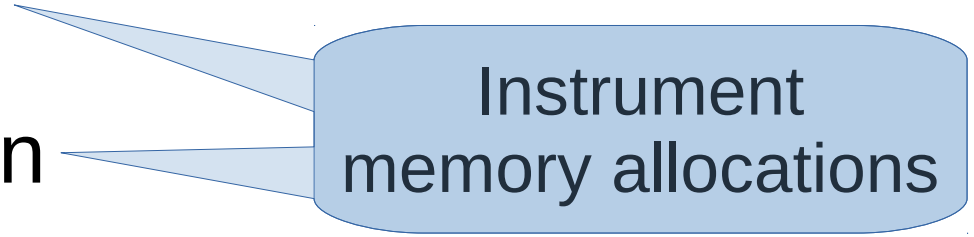
Memory store interface

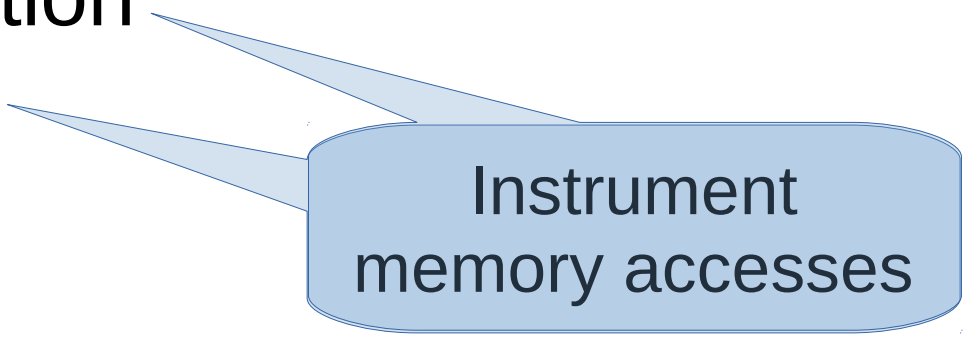
- Allocation
- Deallocation
- Address translation
- Paging system



Memory store interface

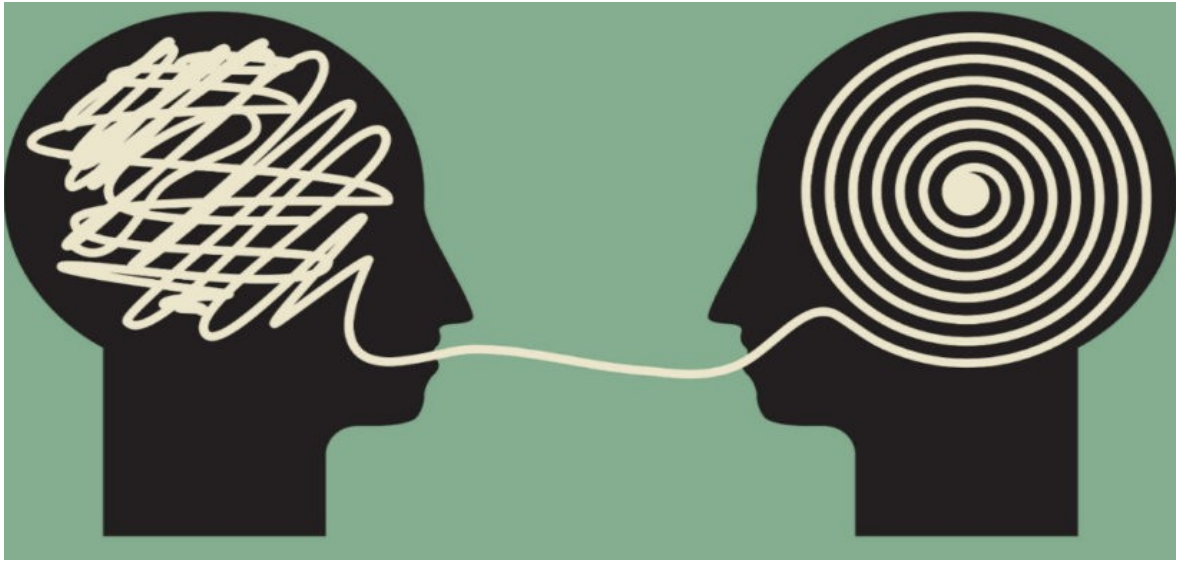
- Allocation
- Deallocation
- Address translation
- Paging system





Software address translation

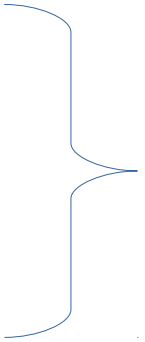
Memory store address Backing store address



More **flexible** than page fault handler customization

Memory store interface

- Allocation
- Deallocation
- Address translation
- Paging system

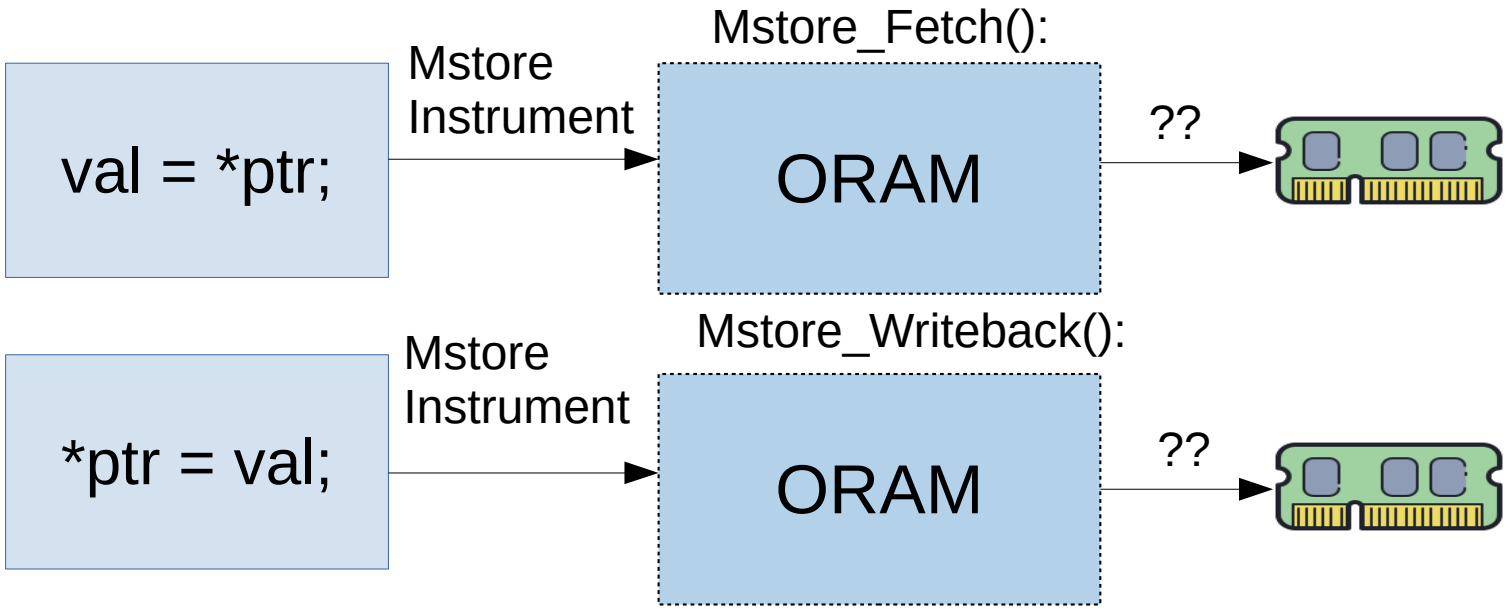


Direct-access Mstore

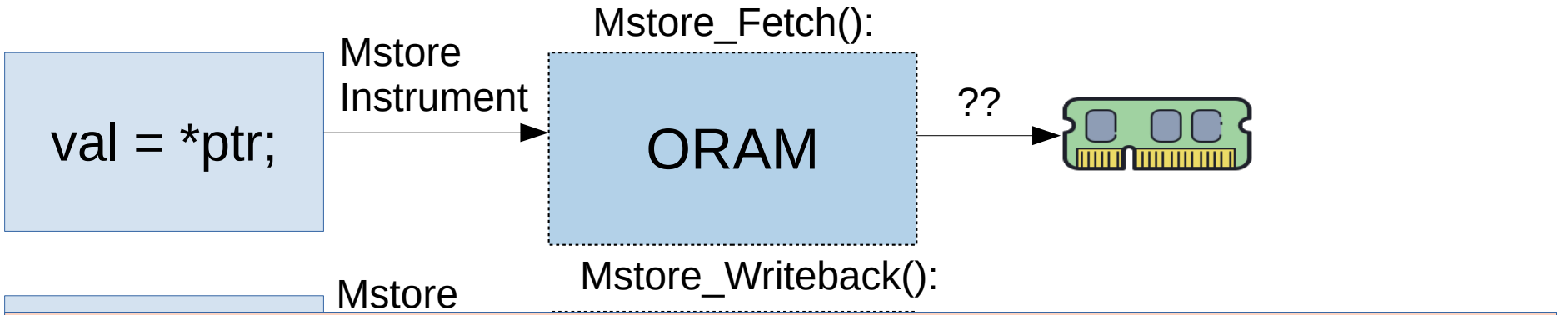
Use case: Oblivious RAM (ORAM)

- Preserves input-output behavior
- Obfuscates distribution of memory accesses

Direct-access memory store



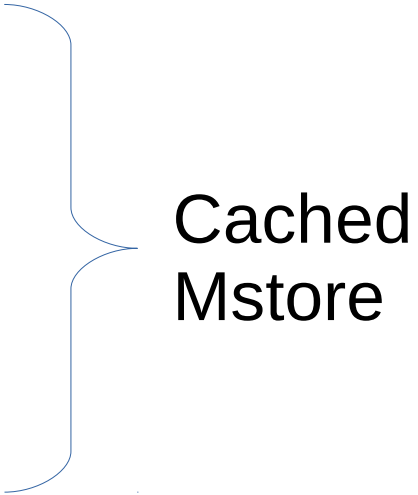
Direct-access memory store



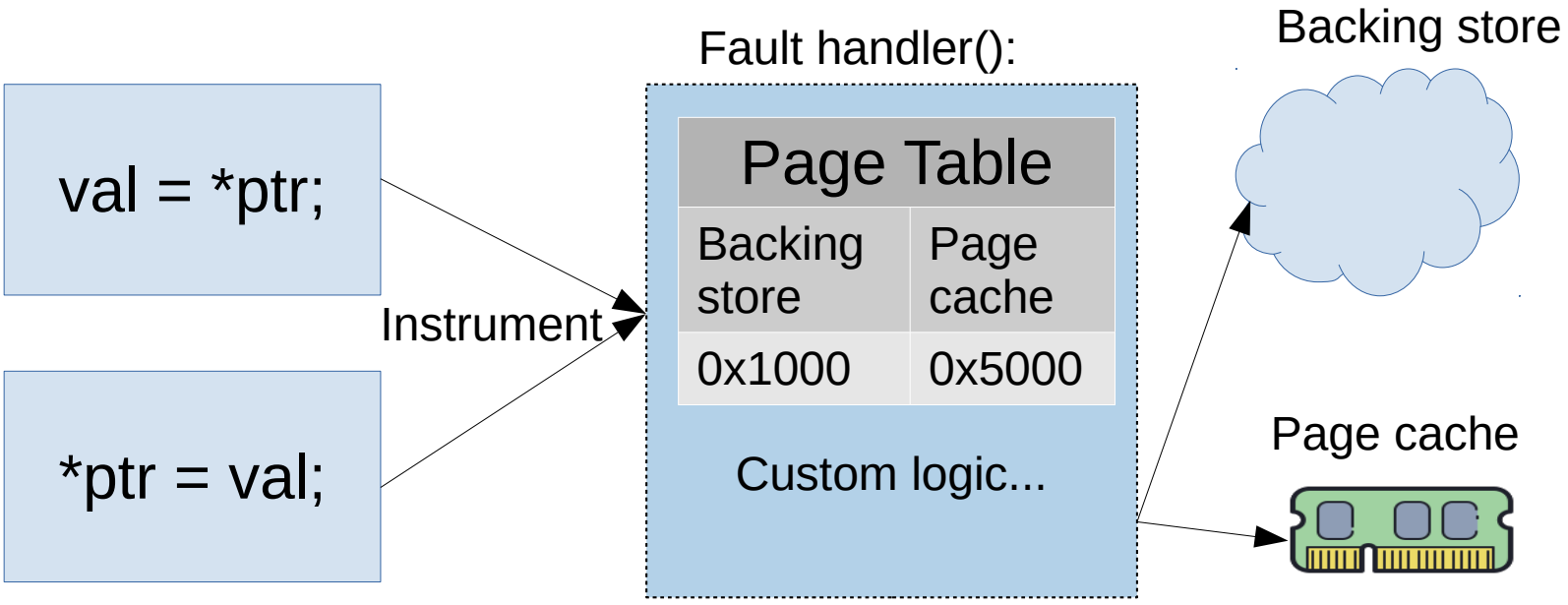
Previous works required manual modifications to use ORAM on every access

Memory store interface

- Allocation
- Deallocation
- Address translation
- Paging system



Cached memory stores

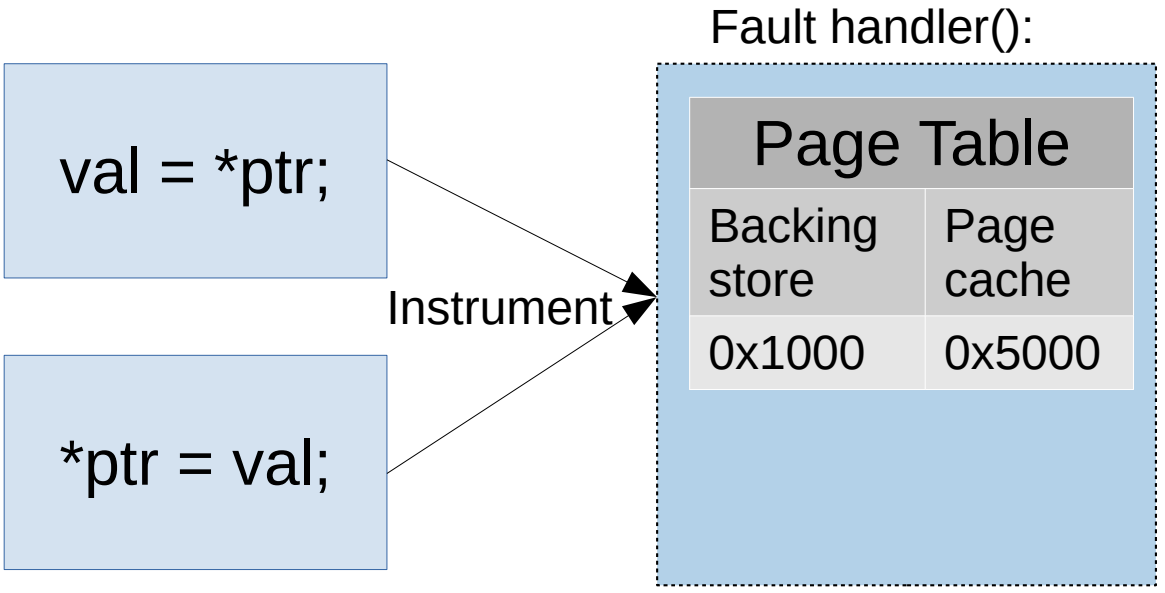


Use case: In-enclave demand-paging

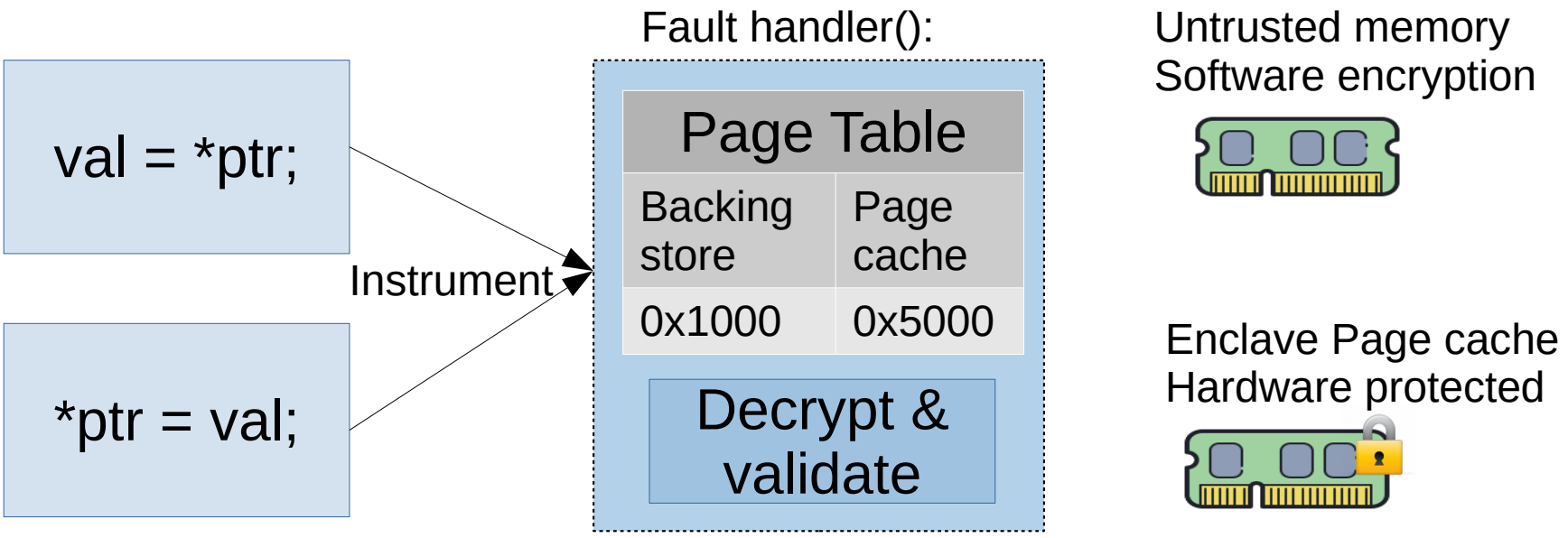
- Maintains SGX enclaves security
- Removes costly enclave exits
  - Boost performance

- Previous work required **manual modifications**

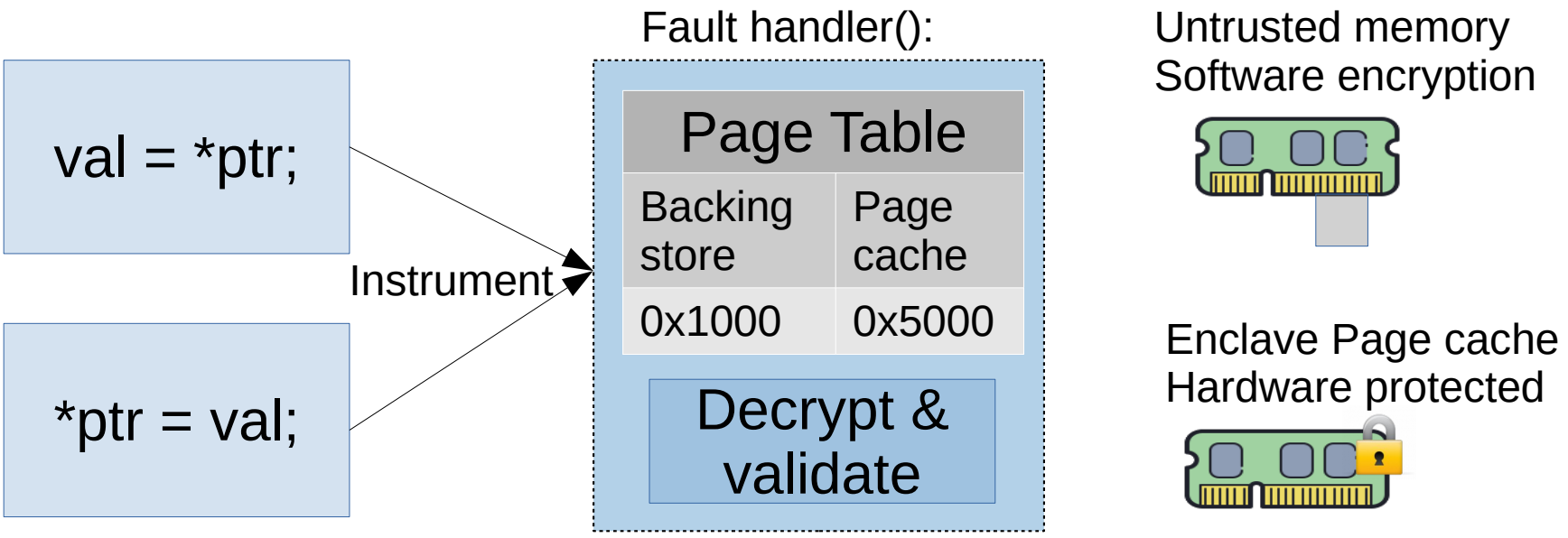
Cached memory stores: **In-enclave demand-paging**



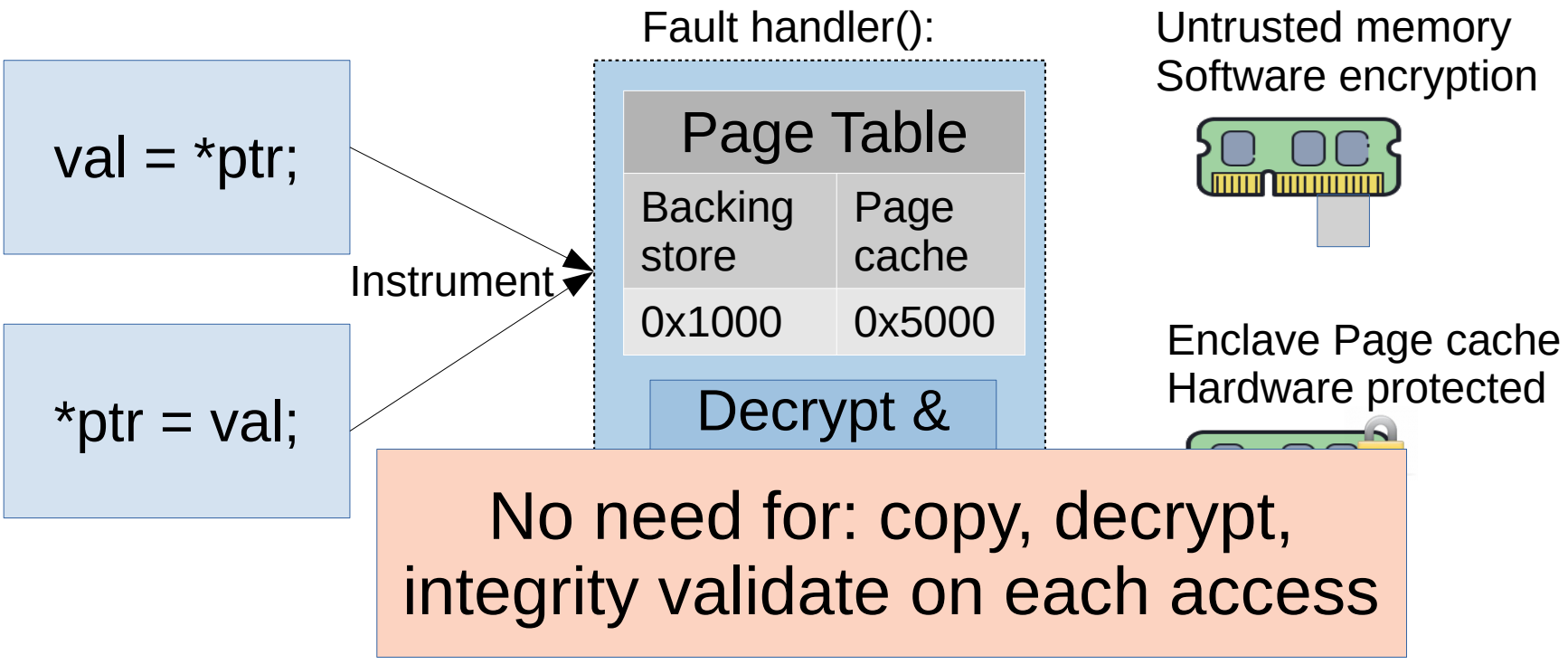
Cached memory stores:
In-enclave demand-paging



Cached memory stores:
In-enclave demand-paging

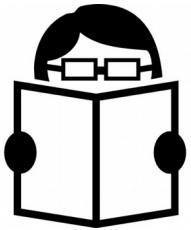


Cached memory stores: **In-enclave demand-paging**



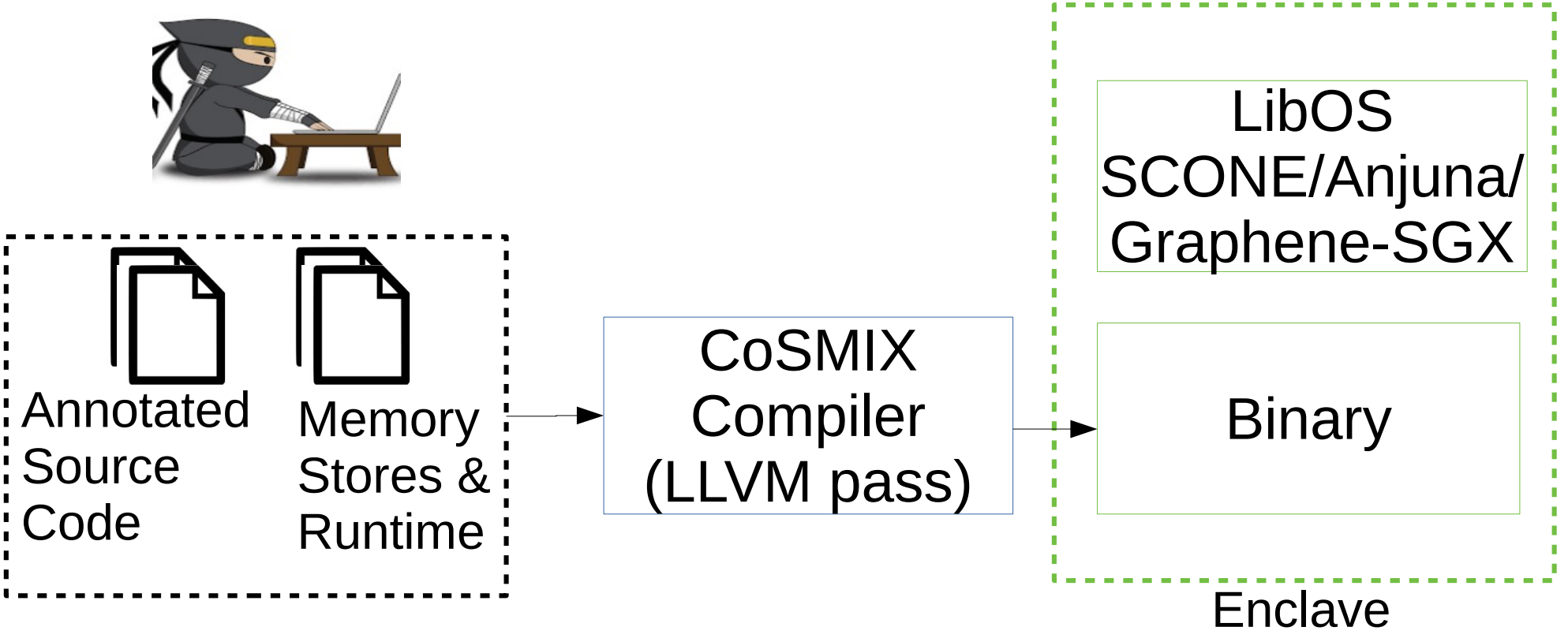
Memory stores are easy

- C/C++
- Small and simple interface: Function callbacks!
- Common building blocks provided



It took a week to implement memory-mapped files mstore

CoSMIX: end-to-end usage



CoSMIX: end-to-end usage

Annotate memory allocations with memory stores

Annotated Source Code

Memory Stores & Runtime

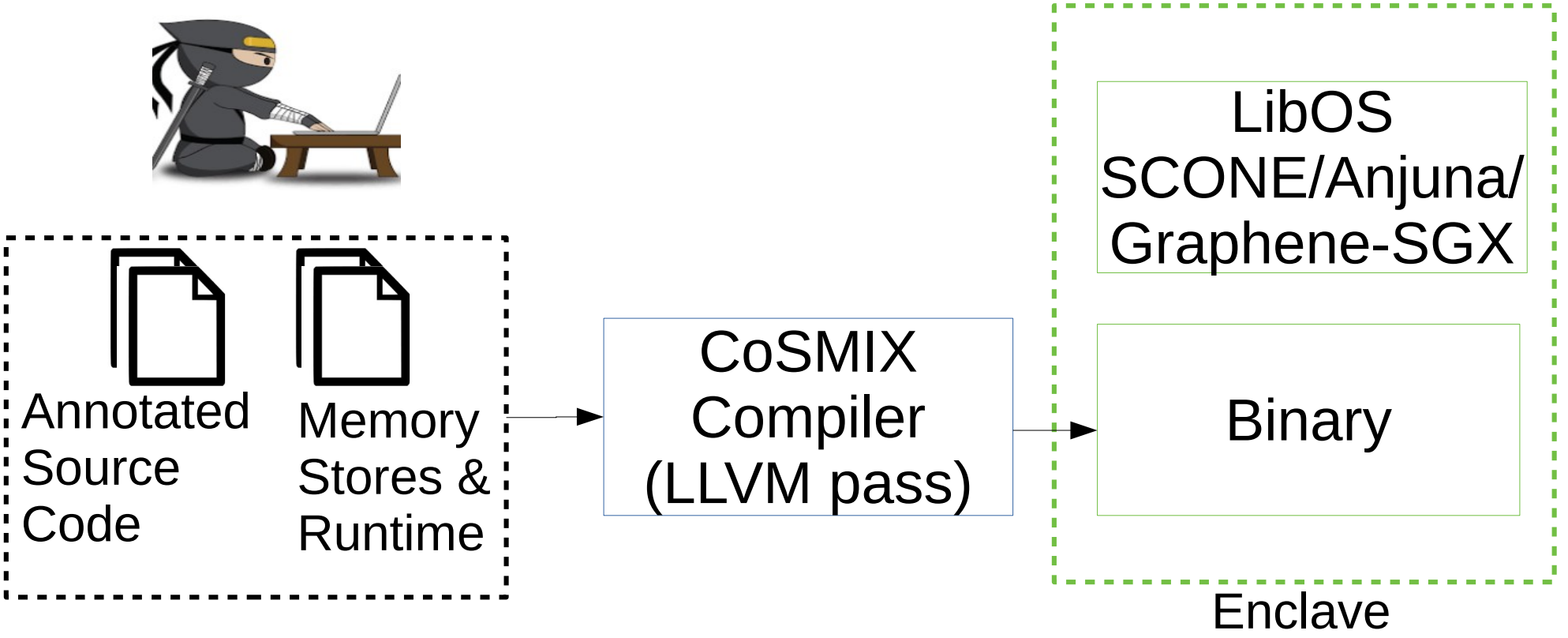
CoSMIX Compiler (LLVM pass)

LibOS SCONE/Anjuna/ Graphene-SGX

Binary

Enclave

CoSMIX: end-to-end usage



CoSMIX: end-to-end usage

Allocations instrumented to infer memory stores' callbacks for each access

Annotated Source Code

Memory Stores & Runtime

CoSMIX Compiler (LLVM pass)

LibOS SCONE/Anjuna/ Graphene-SGX

Binary

Enclave

CoSMIX: end-to-end usage

Allocations instrumented to infer memory stores' callbacks for each access

LibOS
SCONE/Anjuna/
Graphene-SGX

Annotated Source Code

Memory Stor
Runt

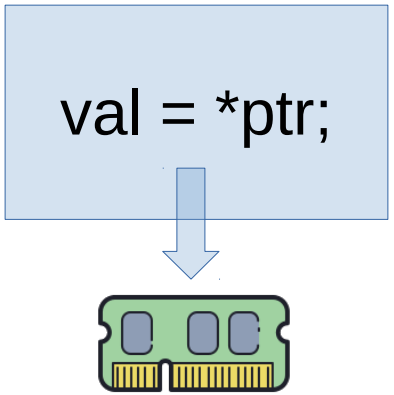
CoSMIX Compiler

Binary

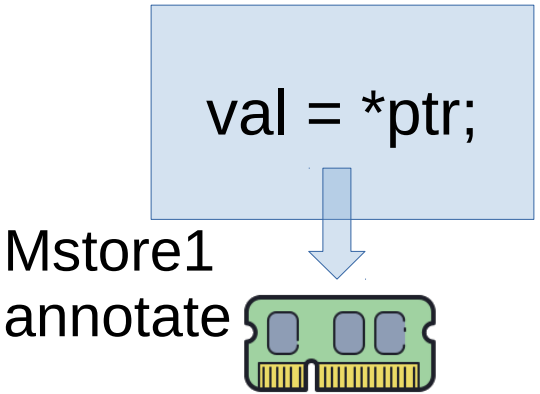
Mix different memory stores!

clave

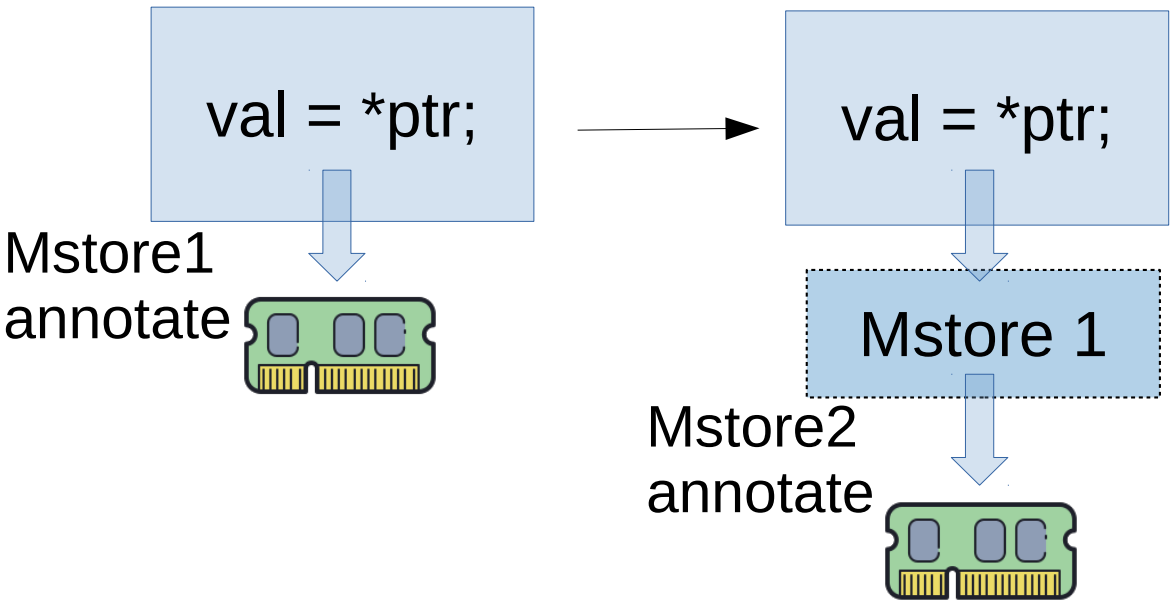
Memory stores are stackable



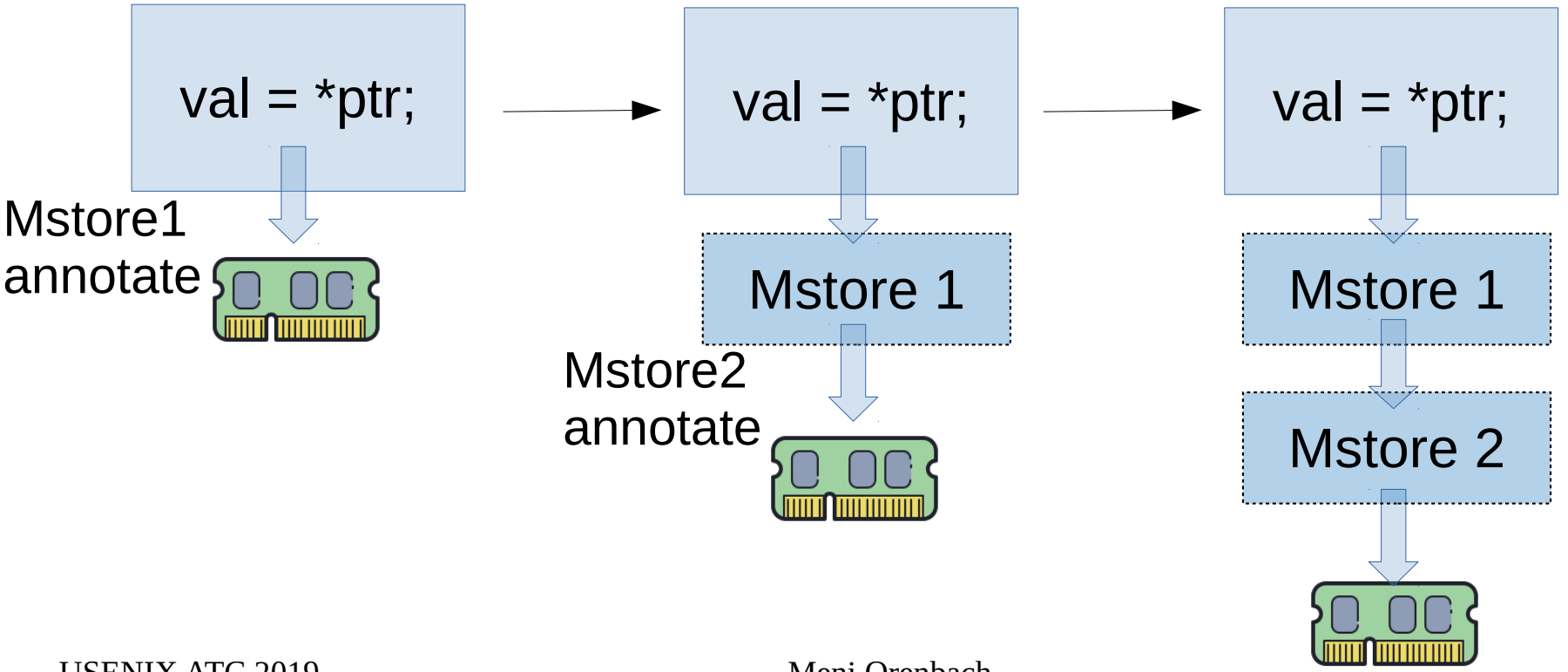
Memory stores are stackable



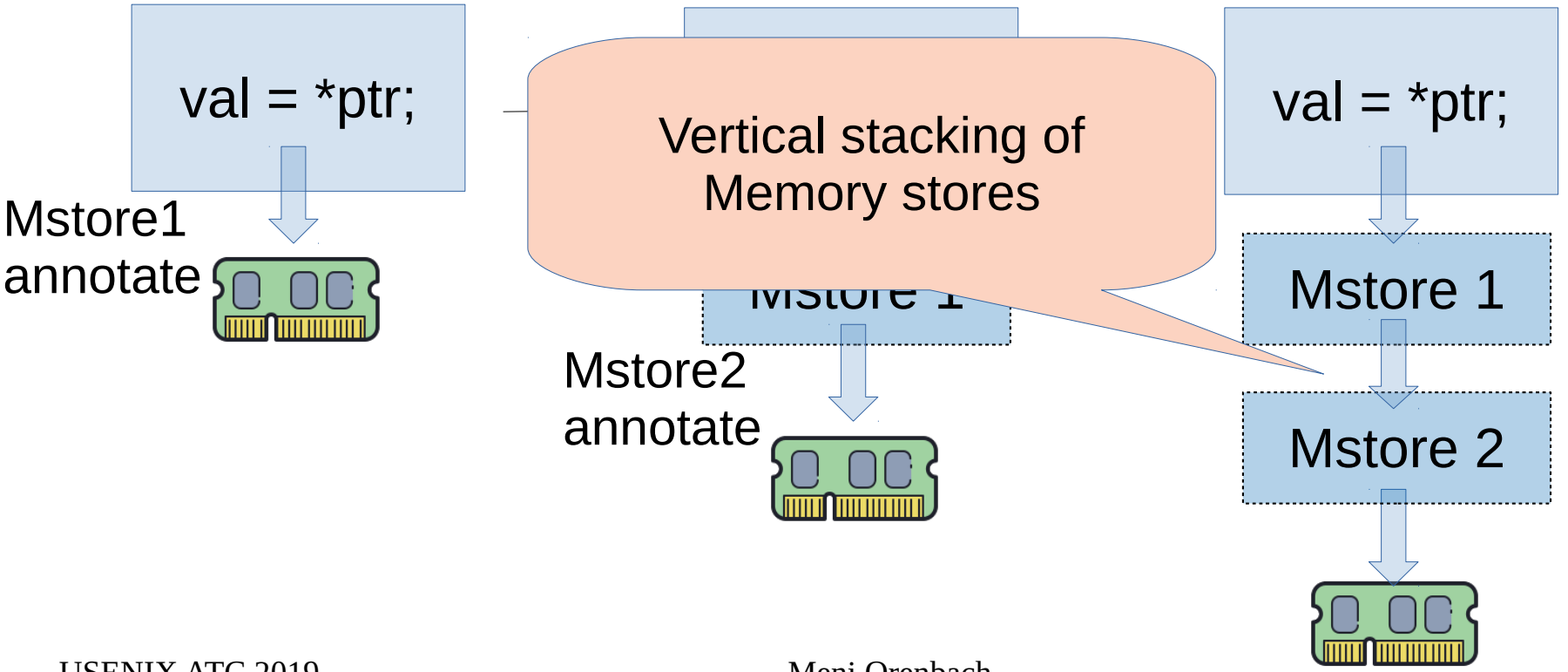
Memory stores are stackable



Memory stores are stackable



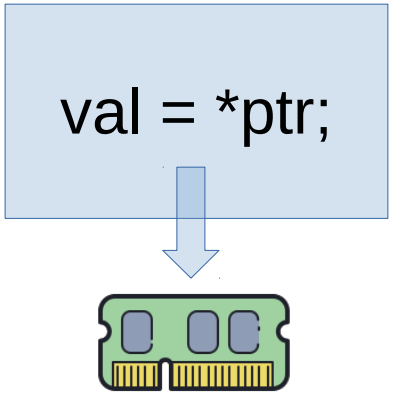
Memory stores are stackable



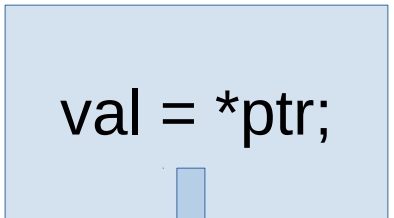
Stacking ORAM on in-enclave paging

- ORAM performs multiple memory accesses to hide memory access patterns
- ORAM data structures increase its memory footprint
- ORAM mstore in SGX may cause **memory thrashing**

Stacking ORAM on in-enclave paging

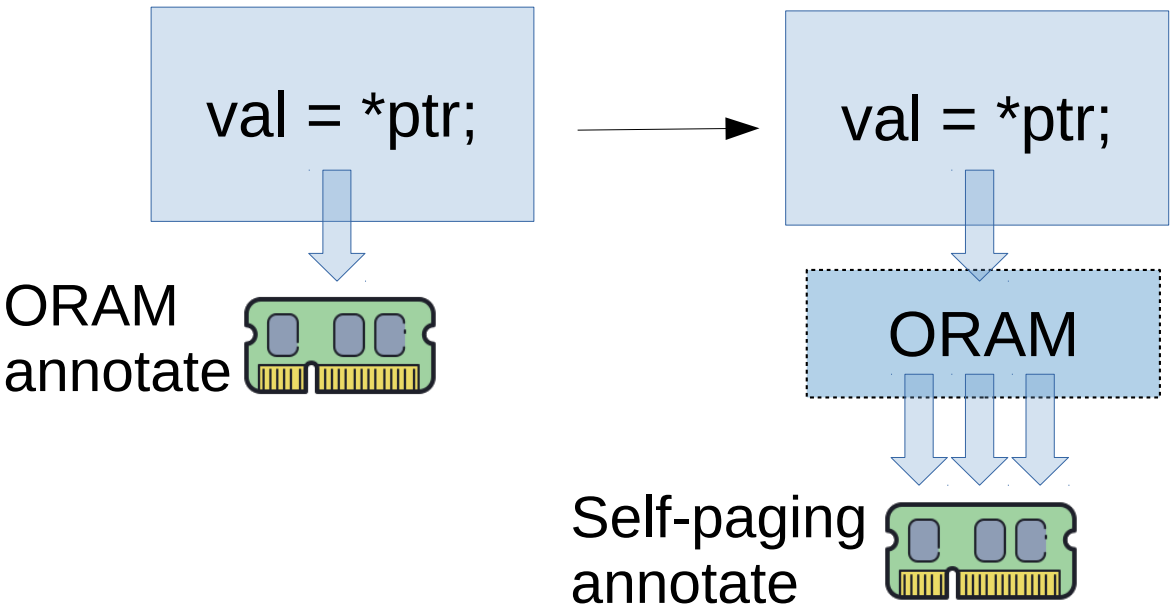


Stacking ORAM on in-enclave paging

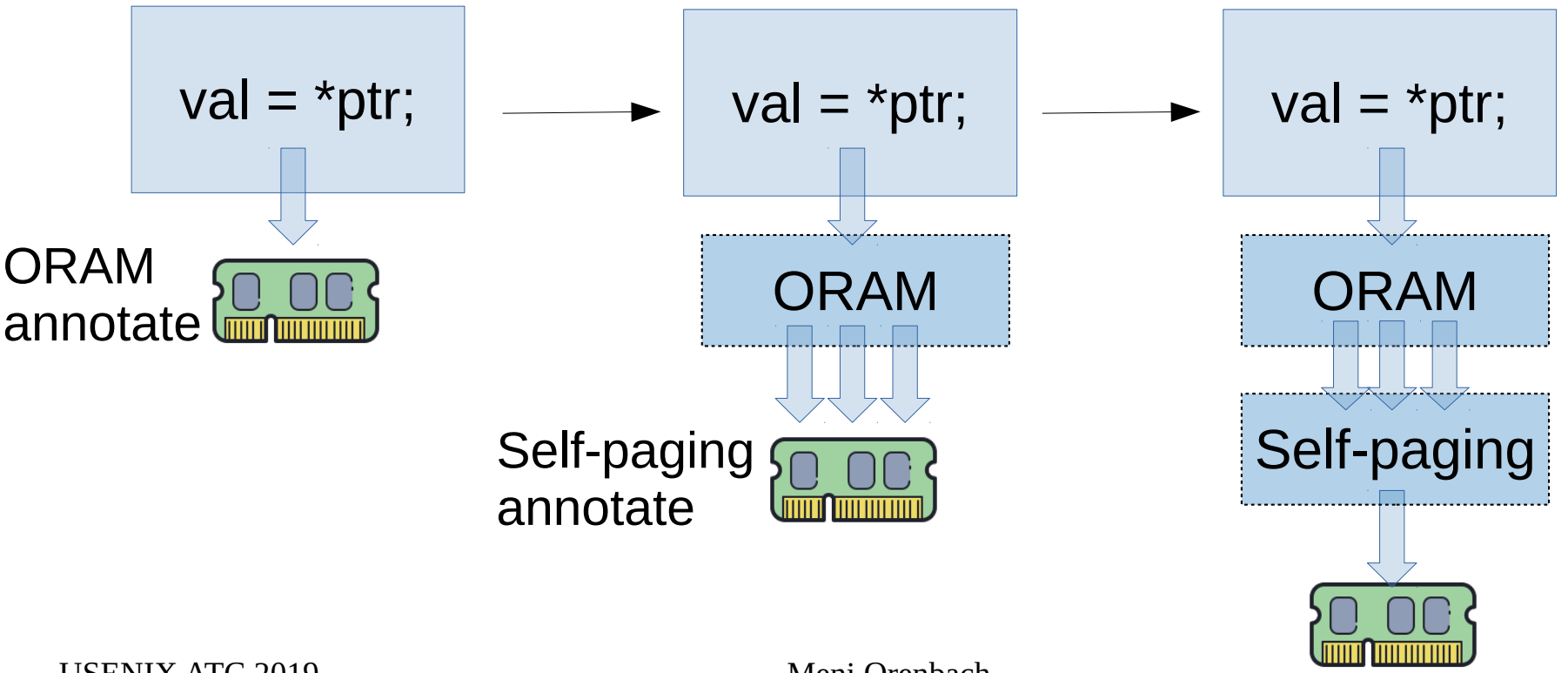


ORAM annotate

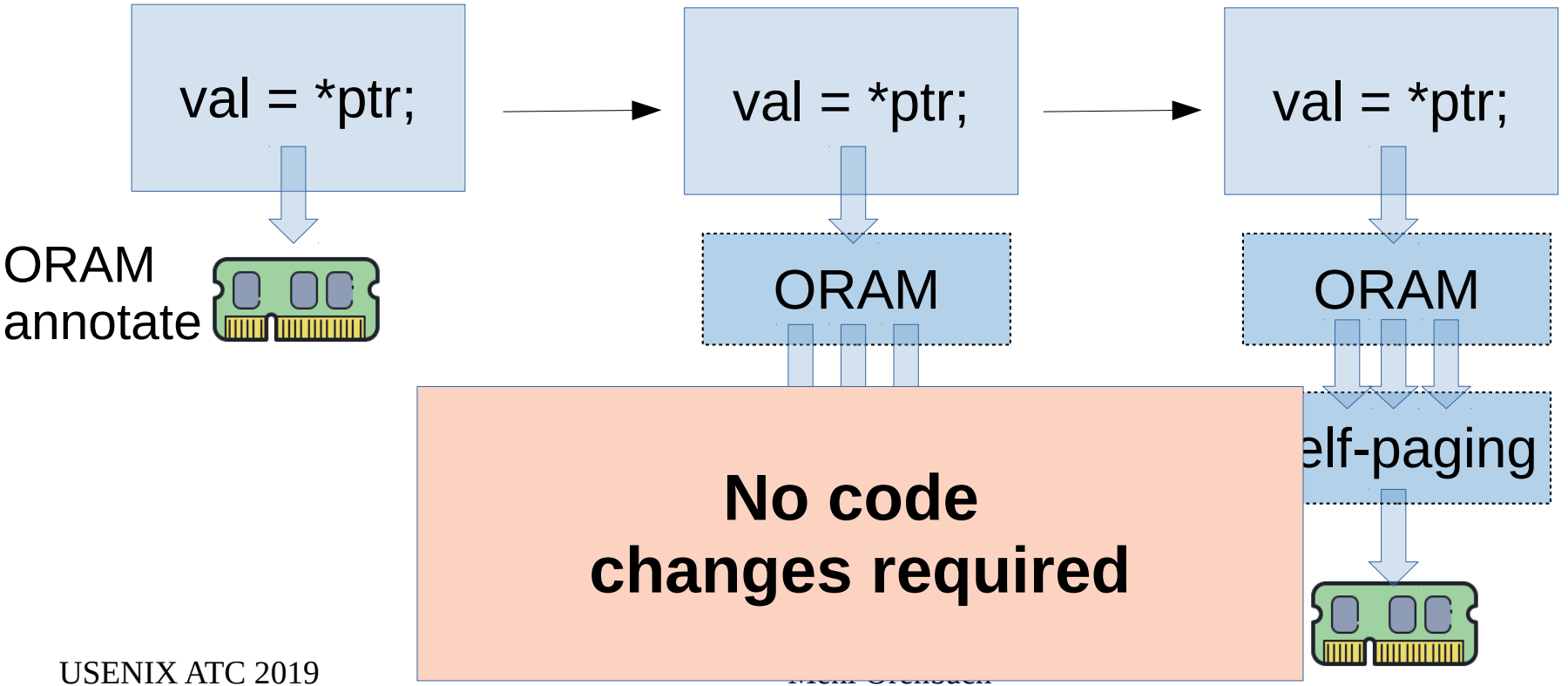
Stacking ORAM on in-enclave paging



Stacking ORAM on in-enclave paging



Stacking ORAM on in-enclave paging

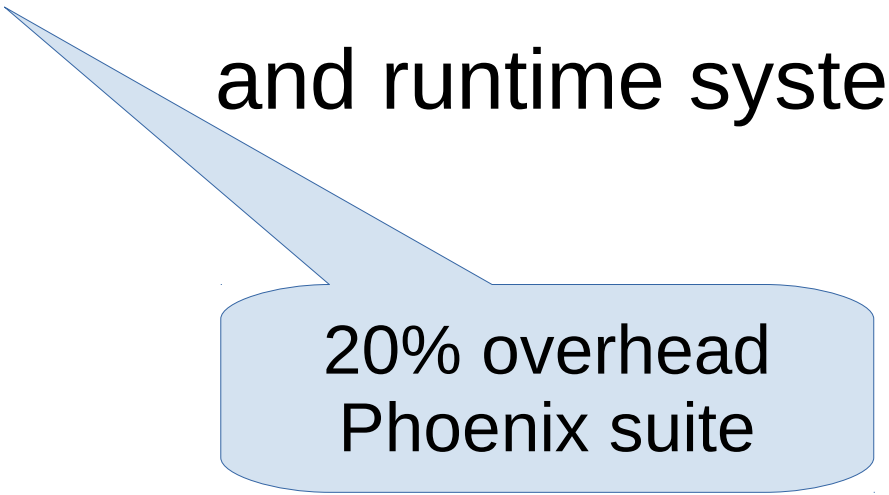


CoSMIX

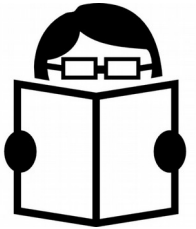
Efficient Instrumenting compiler and runtime system

CoSMIX

Efficient Instrumenting compiler and runtime system



Efficient instrumentation



- **Selective** instrumentation with **pointer analysis**
- Temporal access locality
  - Software TLB: **Cache address translations**
  - Selective instrumentation: **TLB can be small**
- Intensive memory accesses in loops
  - **Hoist** instrumentation for loop-invariant accesses

Agenda

- ~~Motivation~~
- ~~CoSMIX~~
- Evaluation

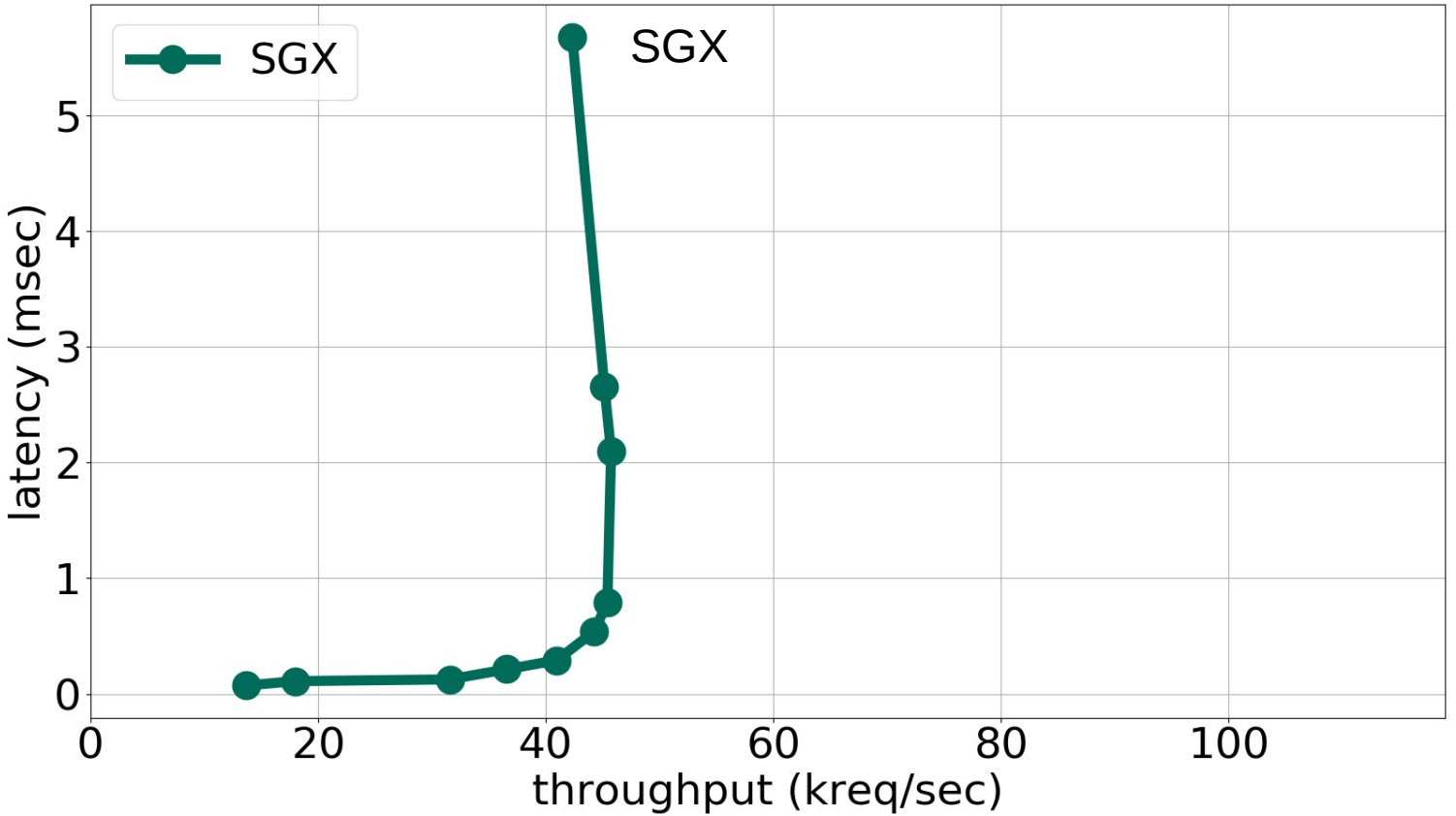
Workloads

Workload	LOC	Added logic
Memcached	16k	In-enclave paging
REDIS	124k	In-enclave paging
SQLITE	135k	Secured file mapping
Face Verification	700	Oblivious page access (ORAM)

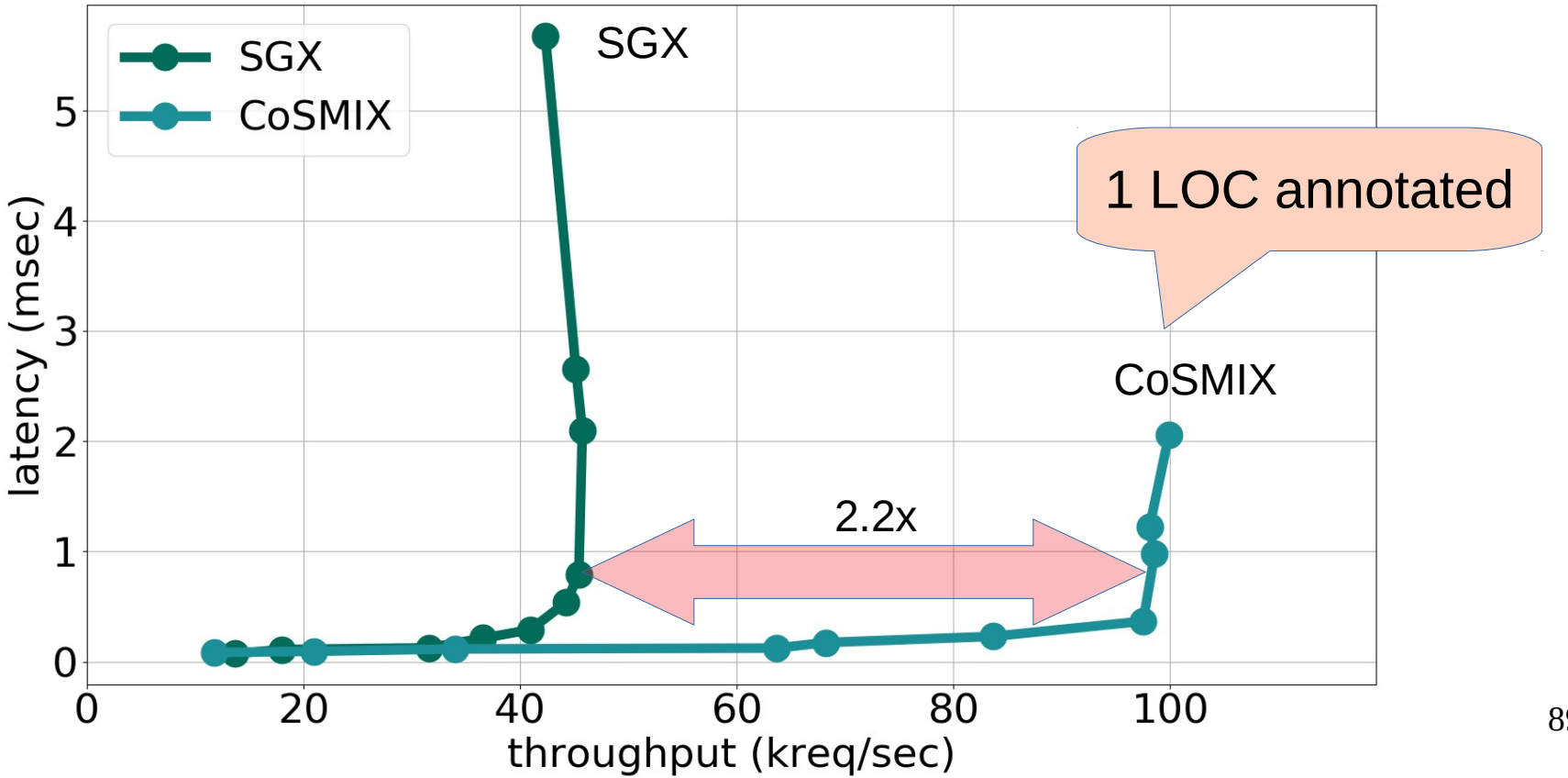
Workloads

Workload	LOC	Added logic
Memcached	16k	In-enclave paging
REDIS	124k	In-enclave paging
SQLITE	135k	Secured file mapping
Face Verification	700	Oblivious page access (ORAM)

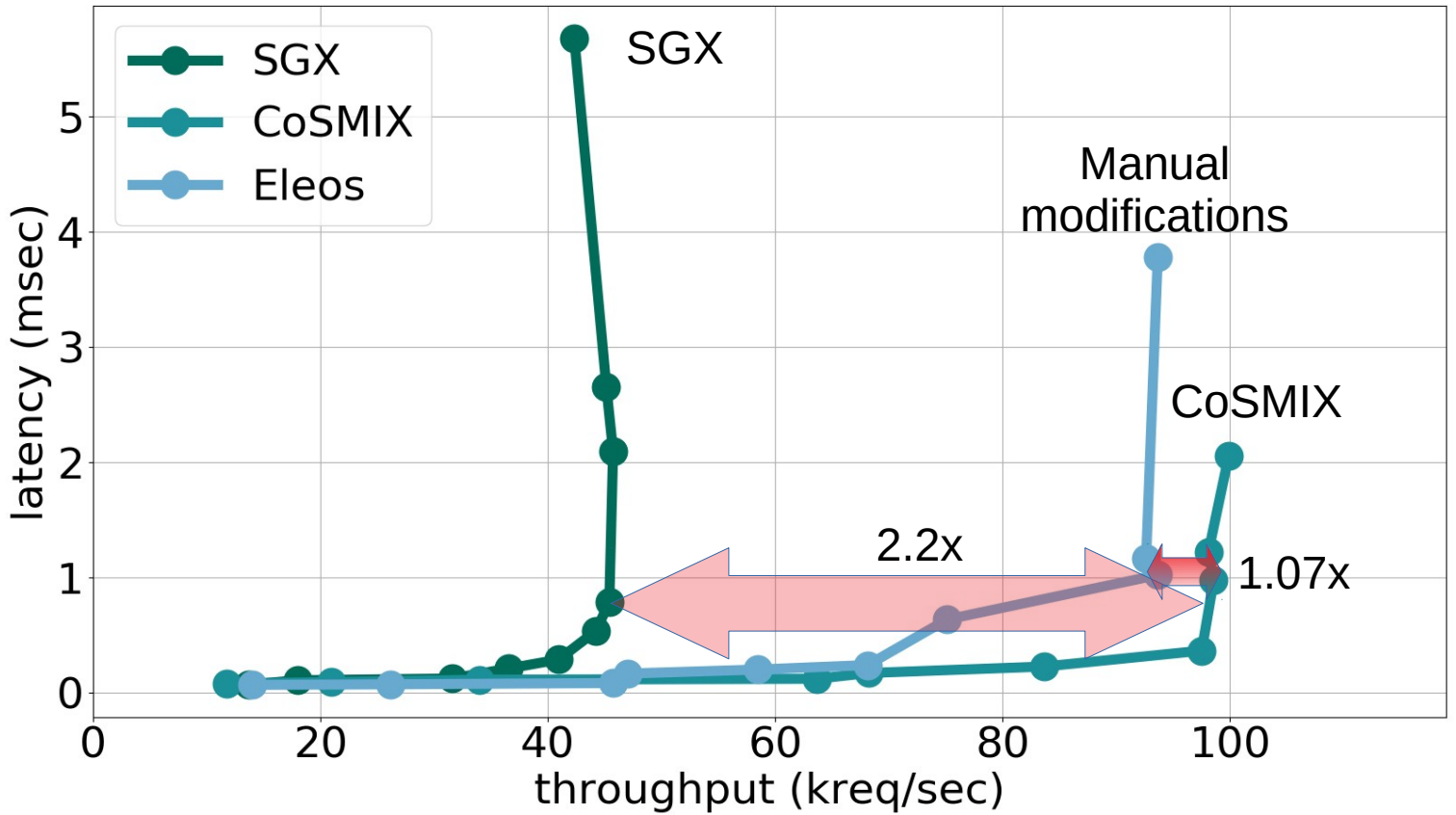
Memcached (600 MB)
Random access 1KB 10% SET



Memcached (600 MB)
Random access 1KB 10% SET

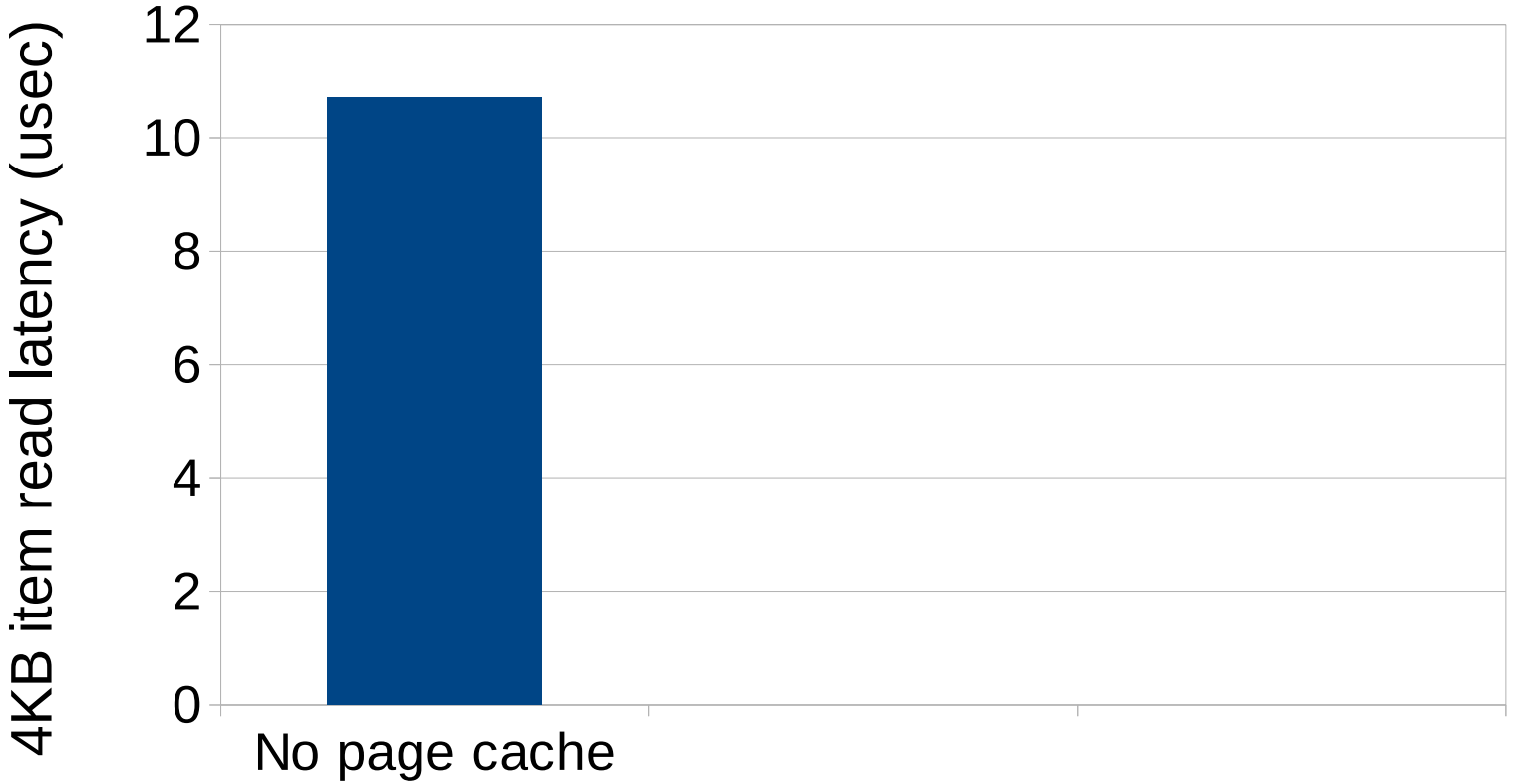


Memcached (600 MB)
Random access 1KB 10% SET



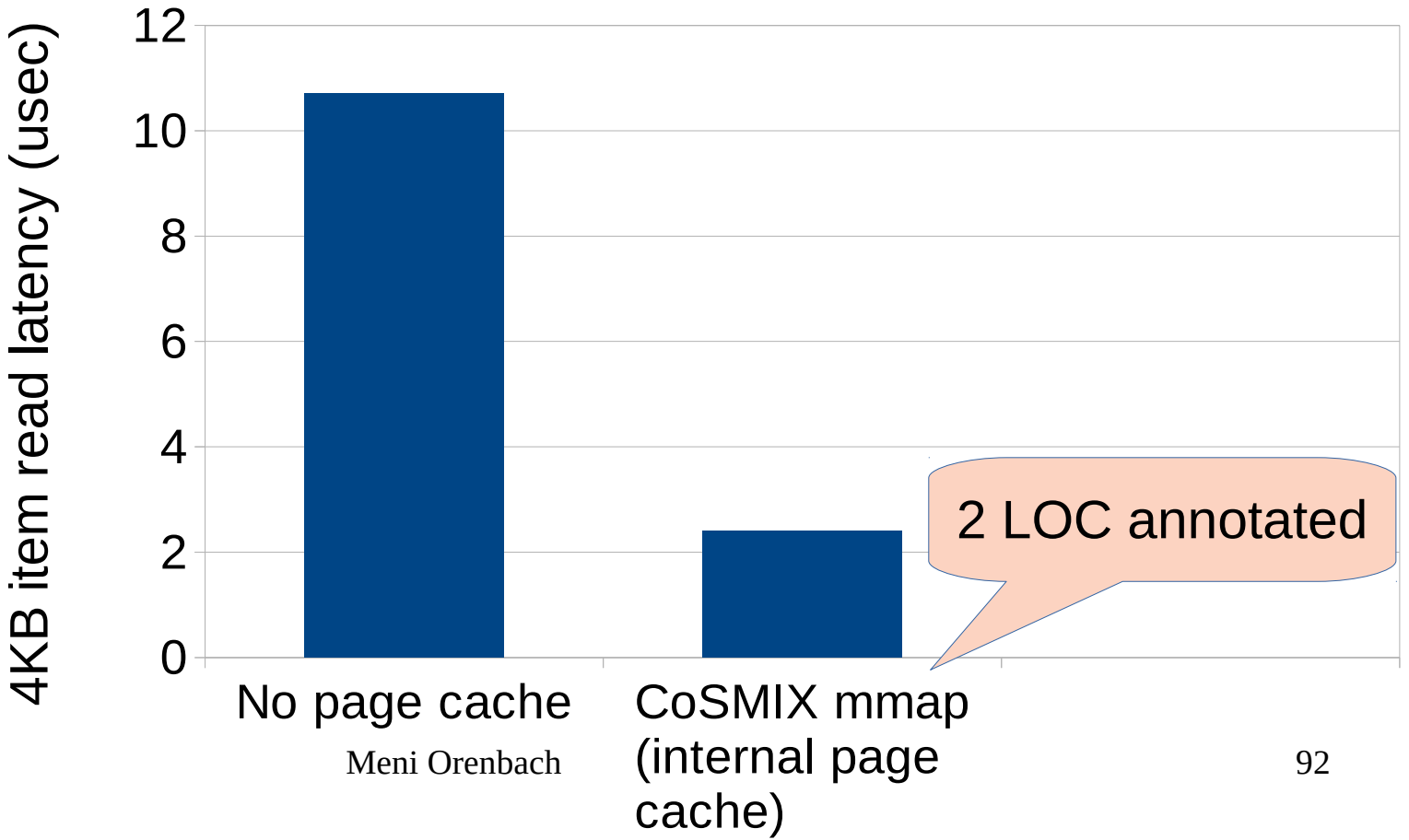
SQLite Encrypted DB file kvtest Random access

Lower is better!



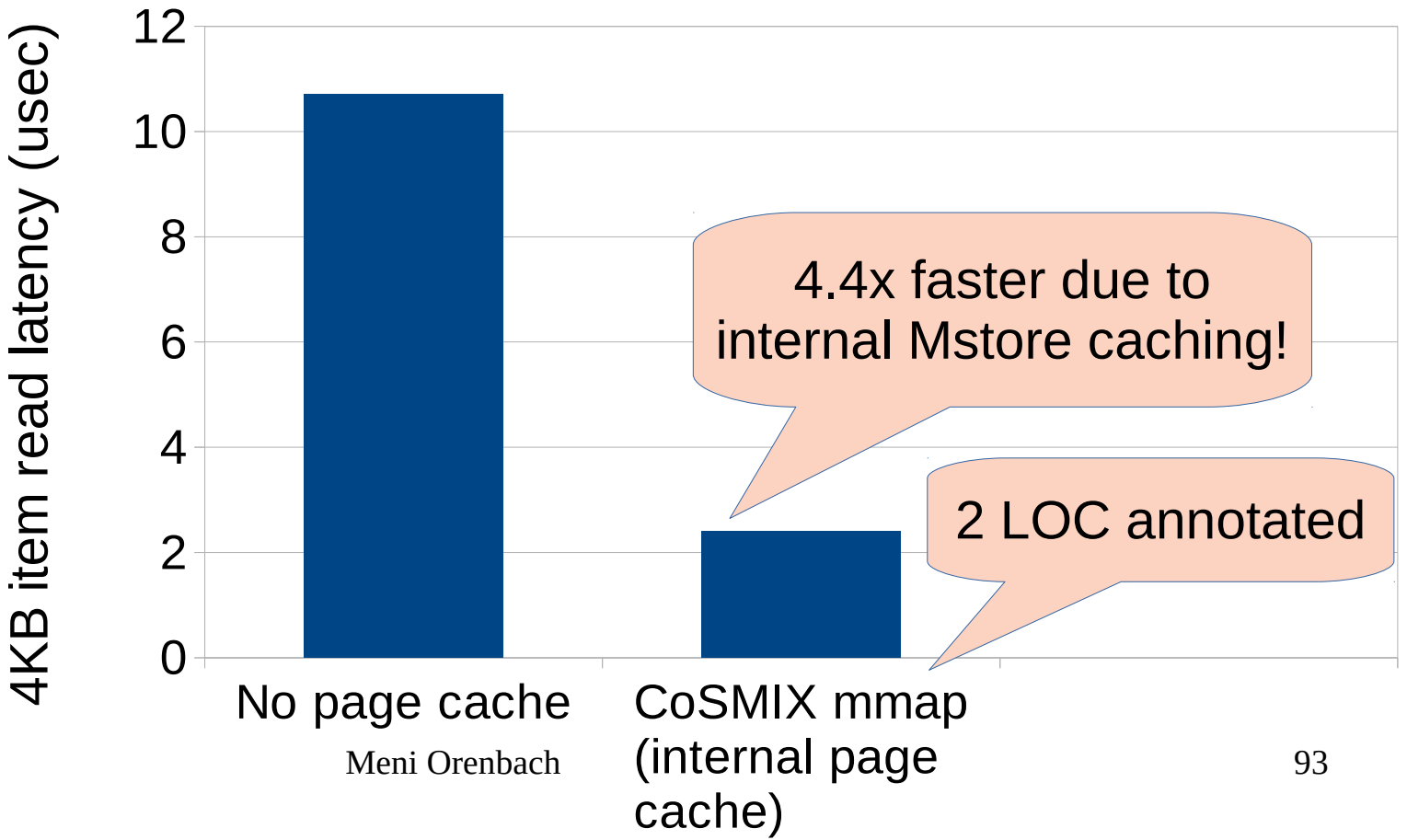
SQLite Encrypted DB file kvtest Random access

Lower is better!



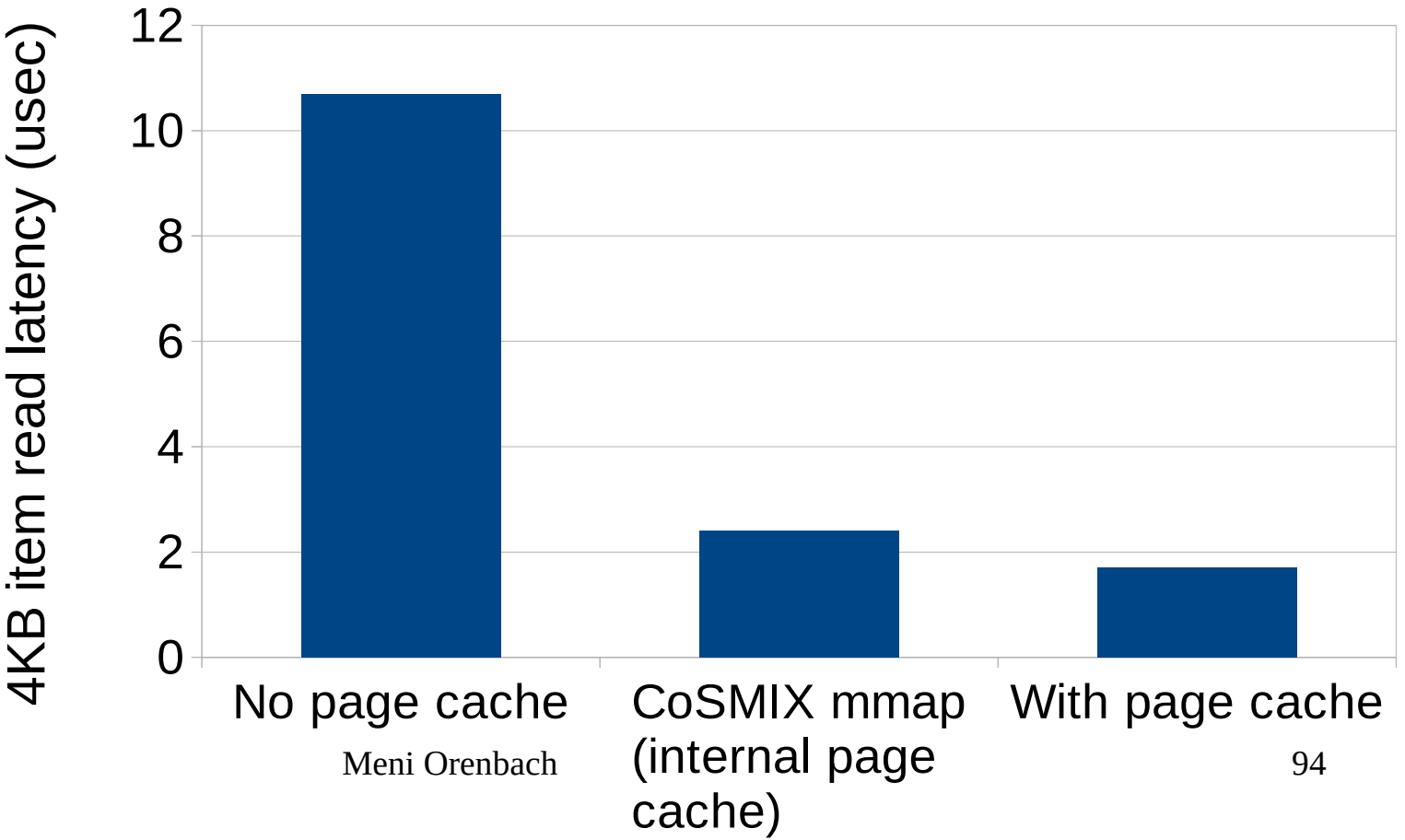
SQLite Encrypted DB file kvtest Random access

Lower is better!



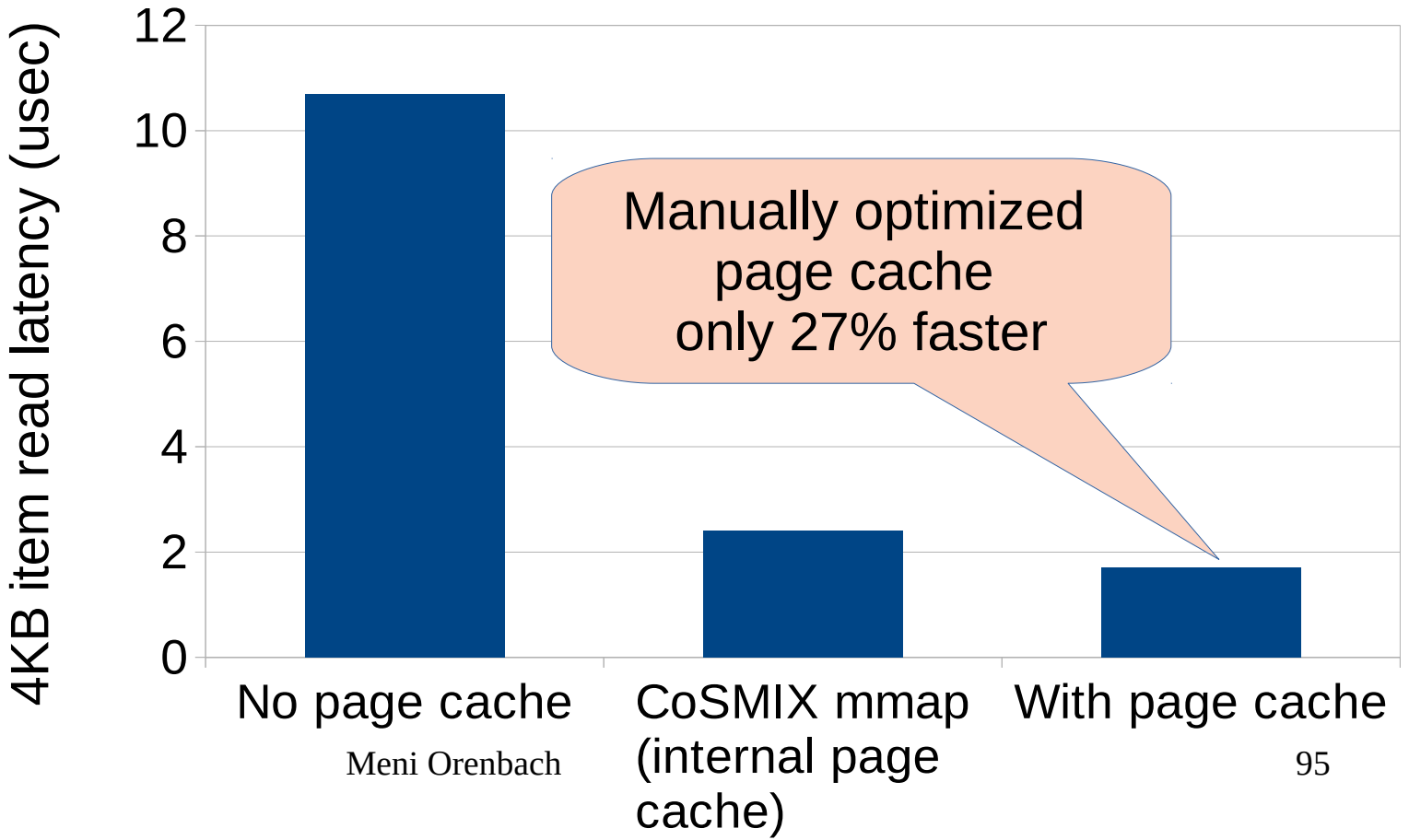
SQLite Encrypted DB file kvtest Random access

Lower is better!



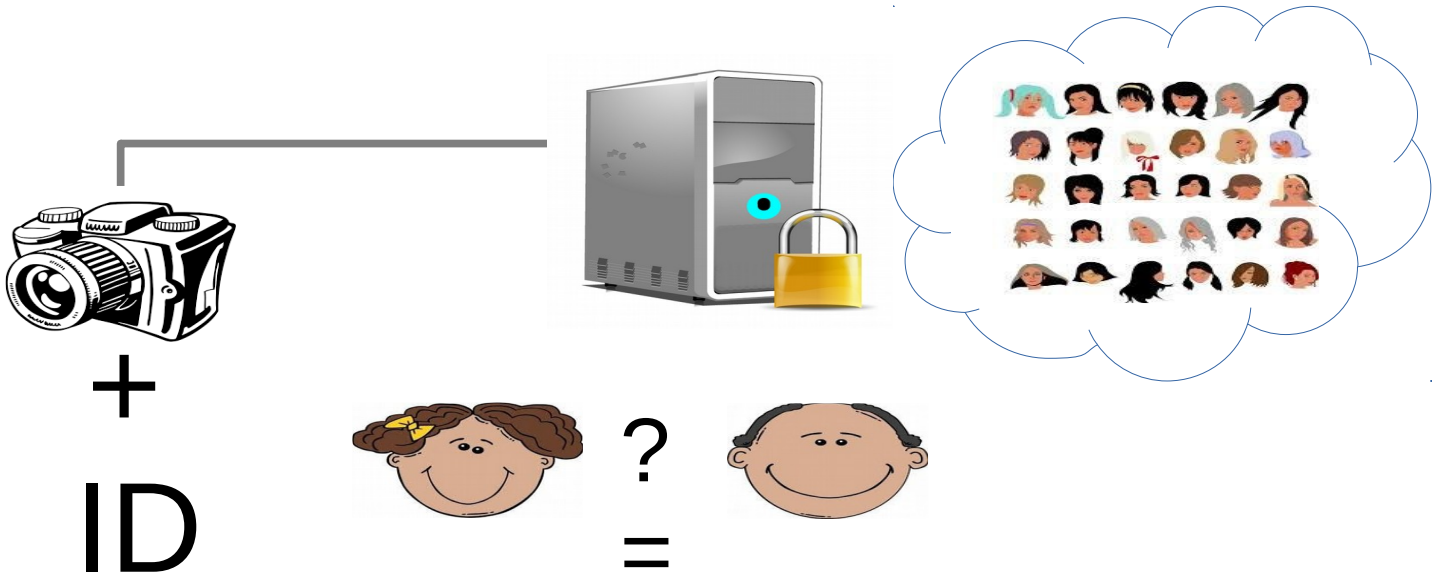
SQLite Encrypted DB file kvtest Random access

Lower is better!

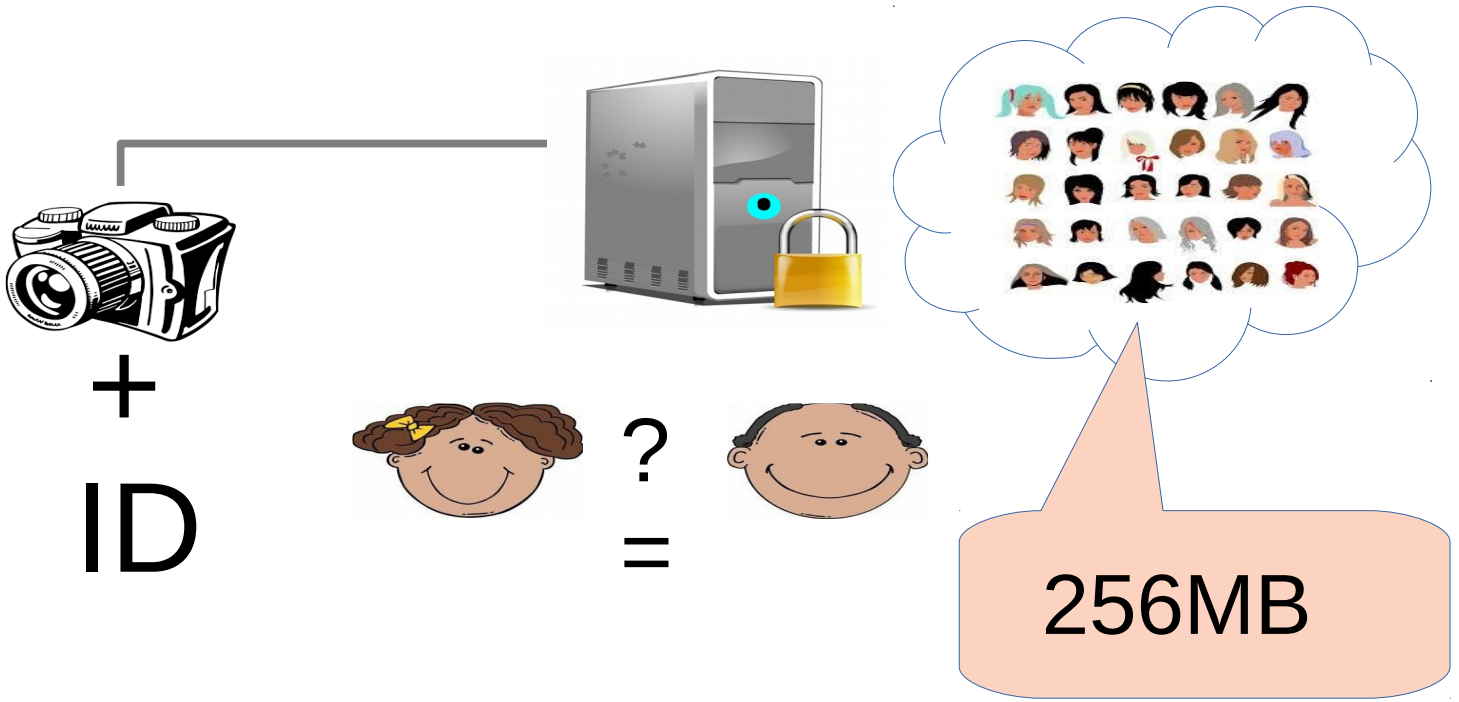


Biometric Identity checking server

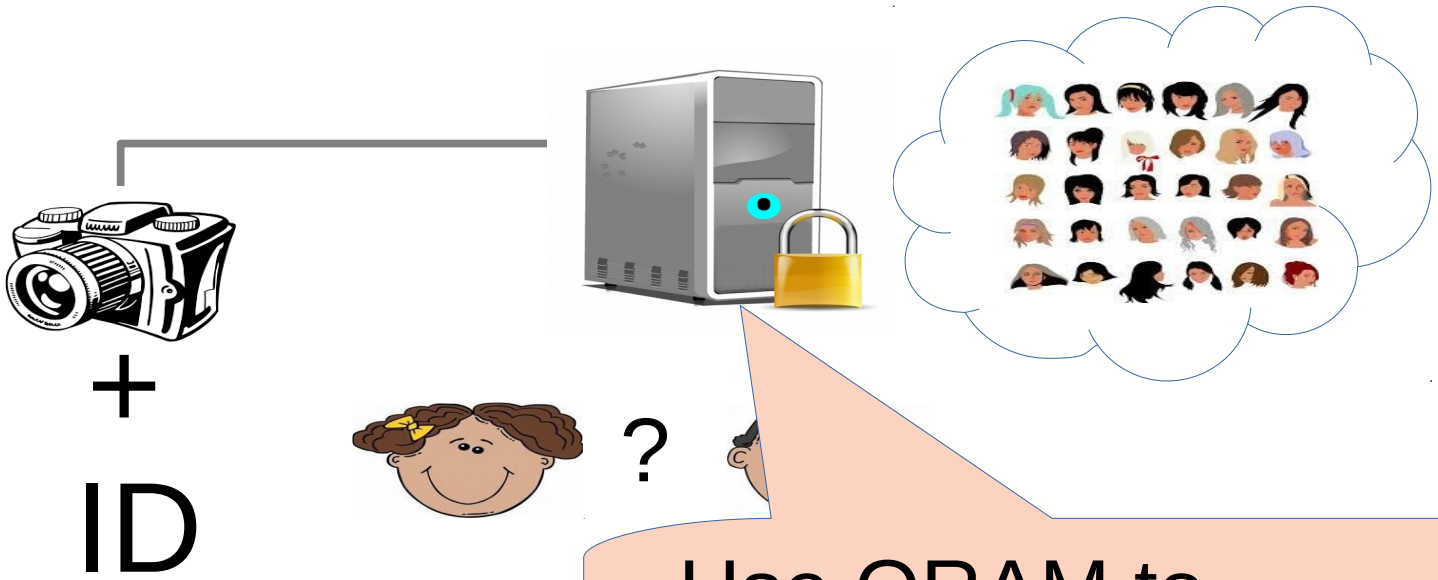
Biometric Identity checking server



Biometric Identity checking server



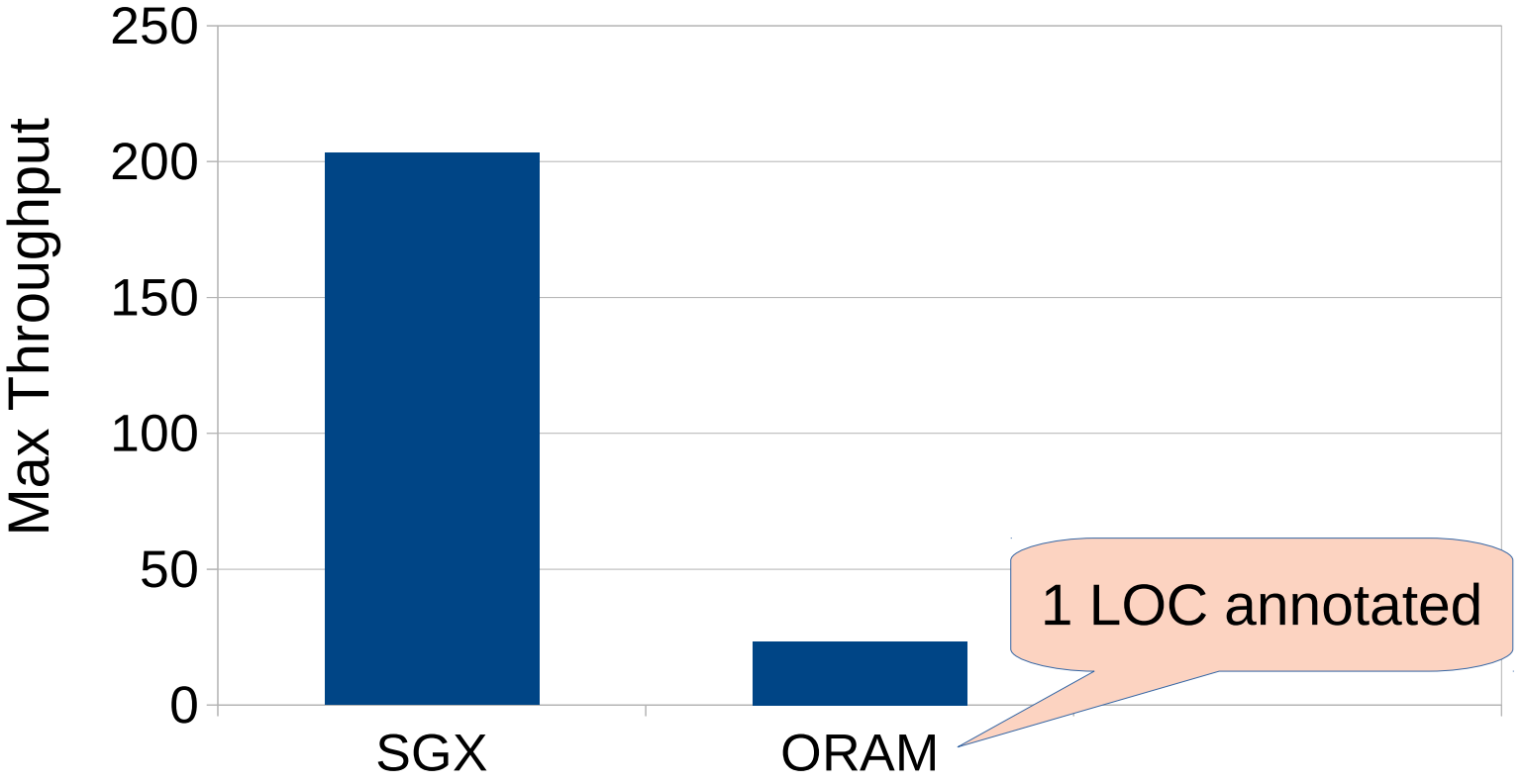
Biometric Identity checking server



ORAM notoriously ineffcient

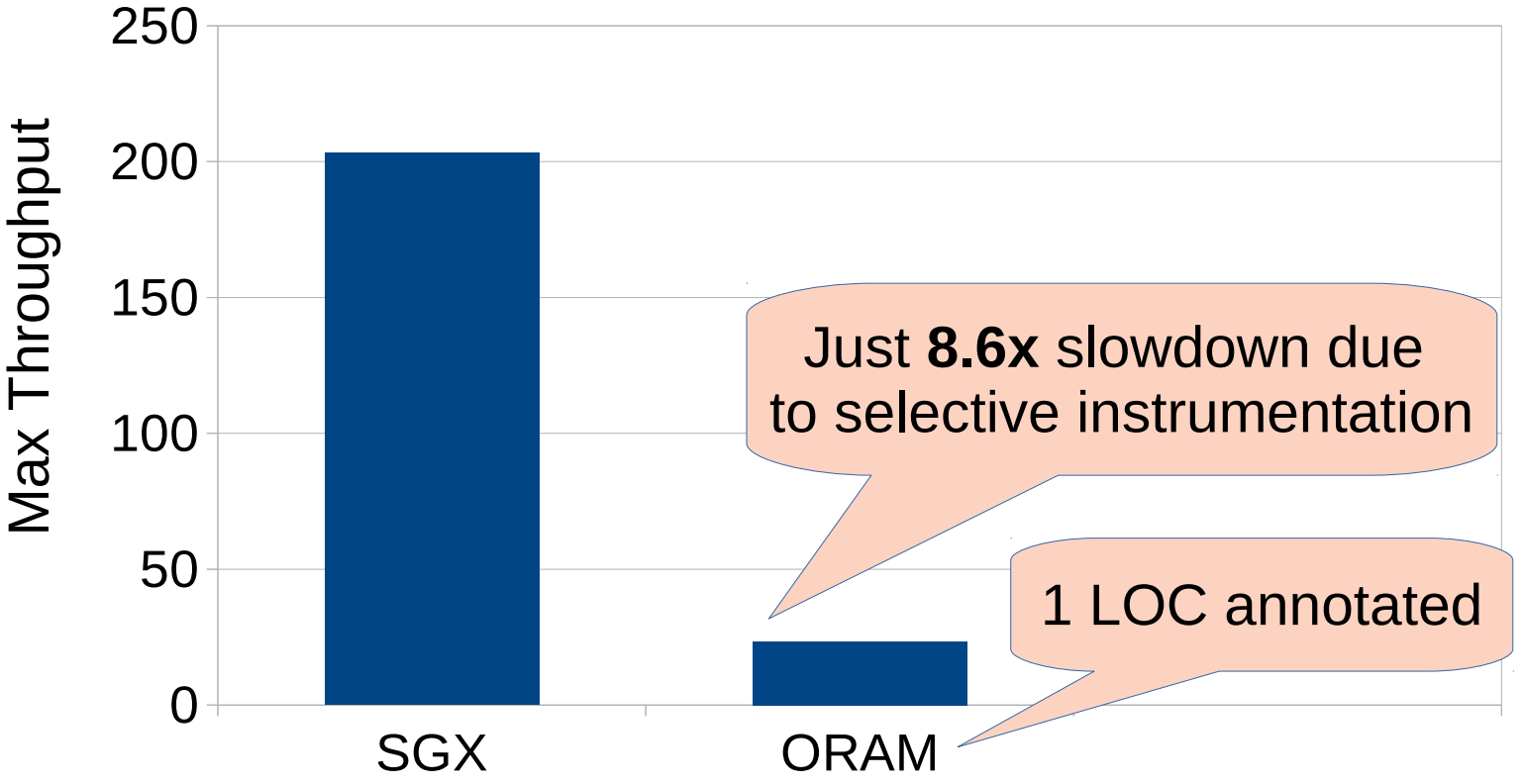
61x slowdown for memory accesses

Face verification



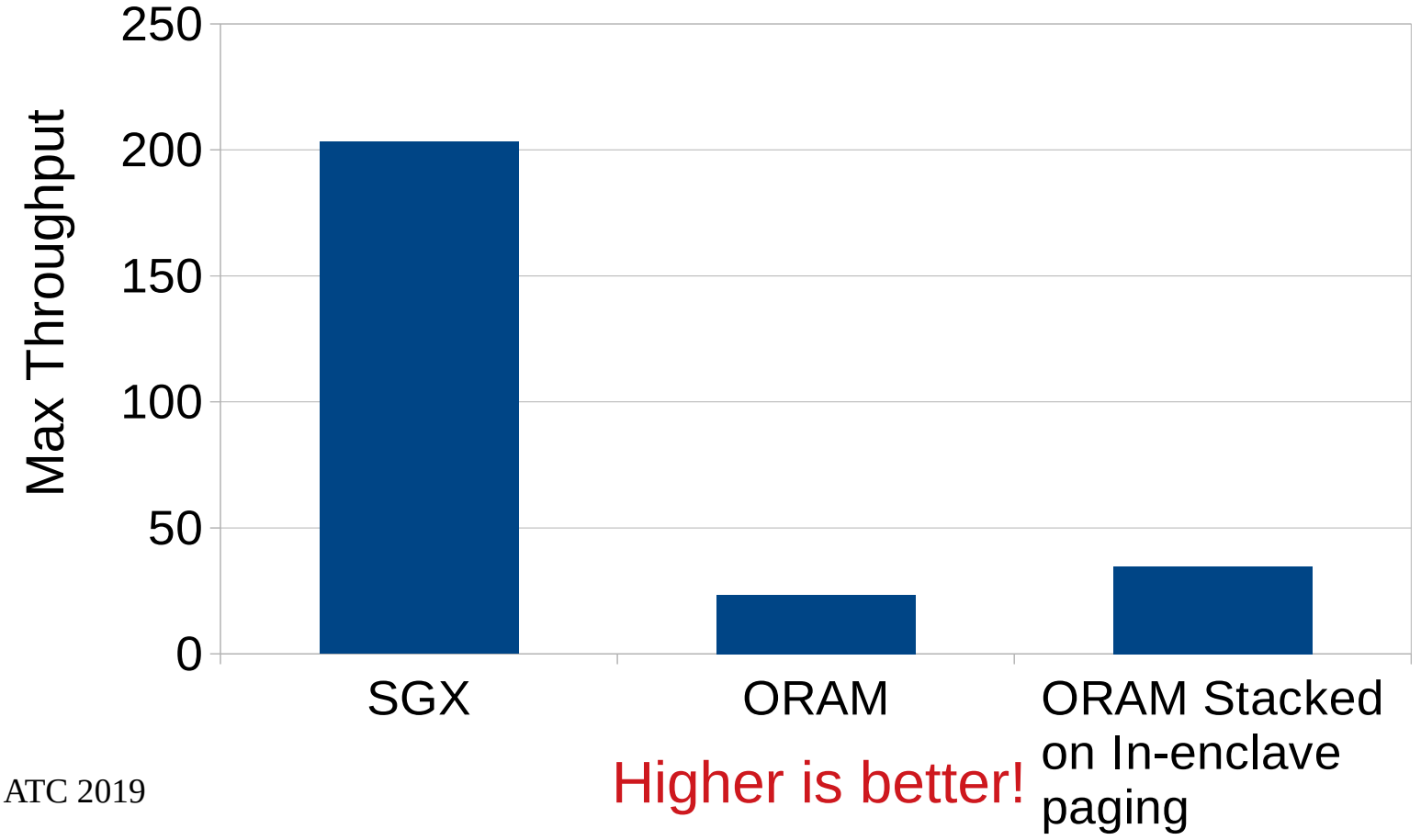
Higher is better!

Face verification

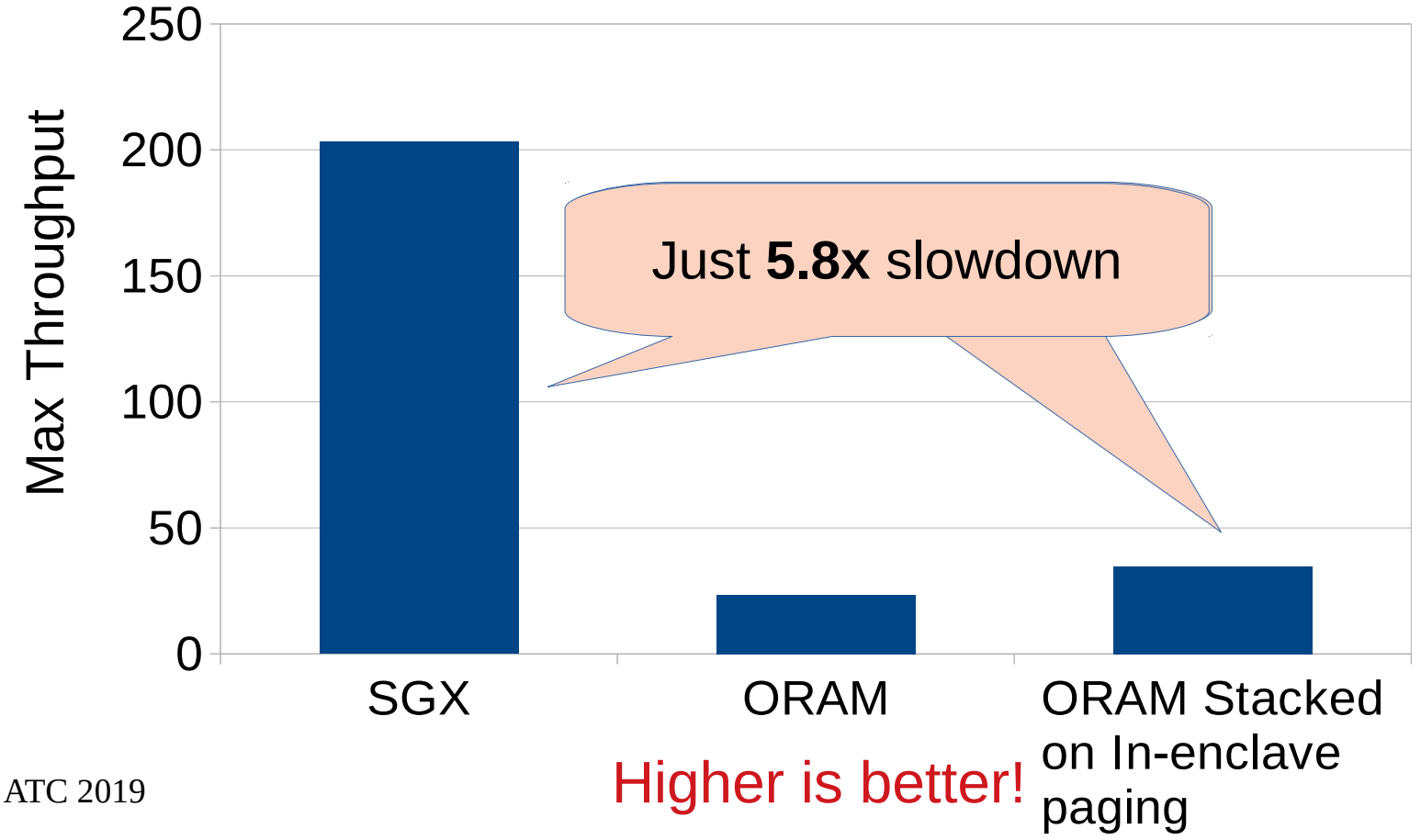


Higher is better!

Face verification



Face verification



Conclusion

- Hardware enclaves improve the security of applications in public cloud environments
- Hardware **limits the use** of a trusted page fault handler
- **CoSMIX** provides trusted page fault handling today
  - Efficient instrumentation makes this a feasible approach

Thank You!

https://github.com/acsl-technion/cosmix