SGXBounds

Memory Safety for Shielded Execution

Dmitrii Kuvaiskii †, Oleksii Oleksenko †, Sergei Arnautov †, Bohdan Trach †,
Pramod Bhatotia *, Pascal Felber ‡, Christof Fetzer †

† TU Dresden, * The University of Edinburgh, ‡ University of Neuchâtel

Security in the Cloud

- **Security** is a key barrier to adoption of cloud computing

Google Cloud Platform

amazon
web services

Microsoft Azure

Security in the Cloud

- **Security** is a key barrier to adoption of cloud computing
- Attackers compromise **confidentiality** and **integrity**

Google Cloud Platform

amazon
web services

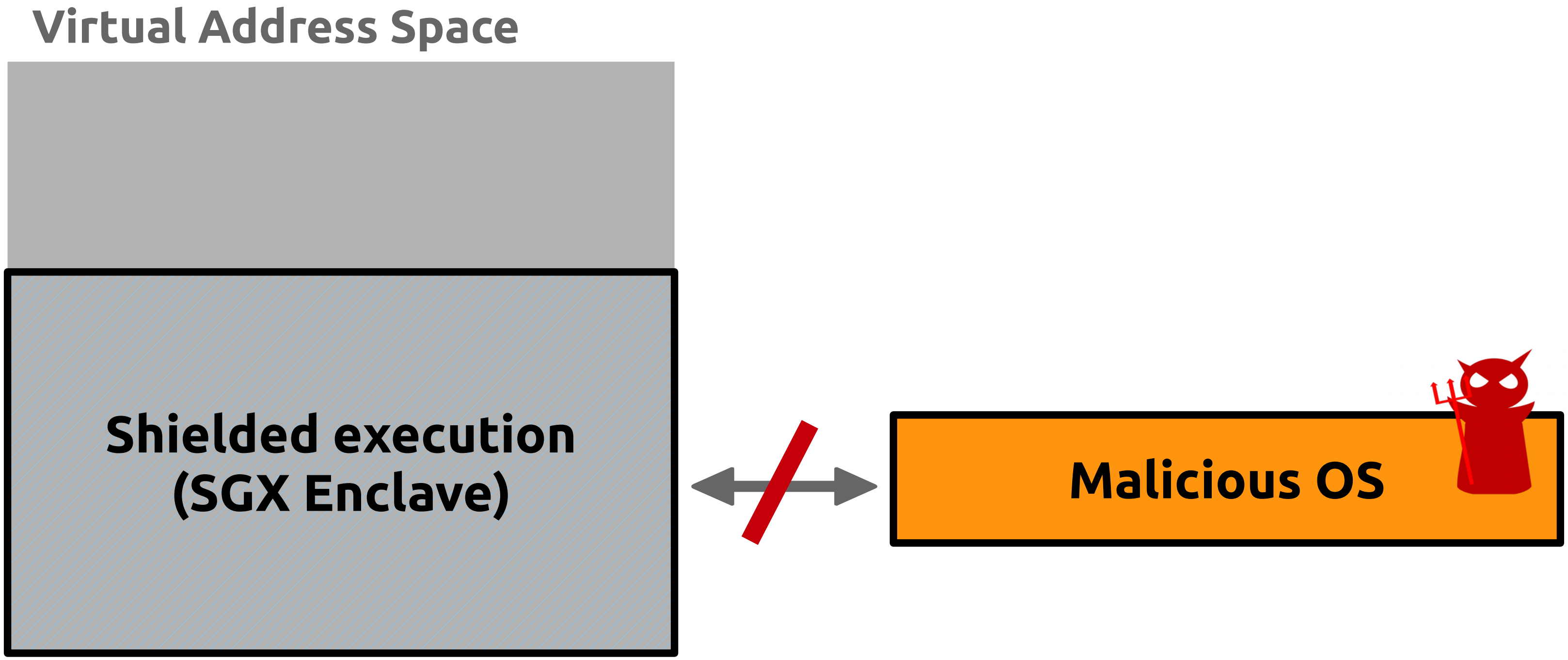
Microsoft Azure

Security in the Cloud

- **Security** is a key barrier to adoption of cloud computing
- Attackers compromise **confidentiality** and **integrity**
  - ➡ Malicious host (e.g., cloud provider)
  - ➡ Software vulnerabilities

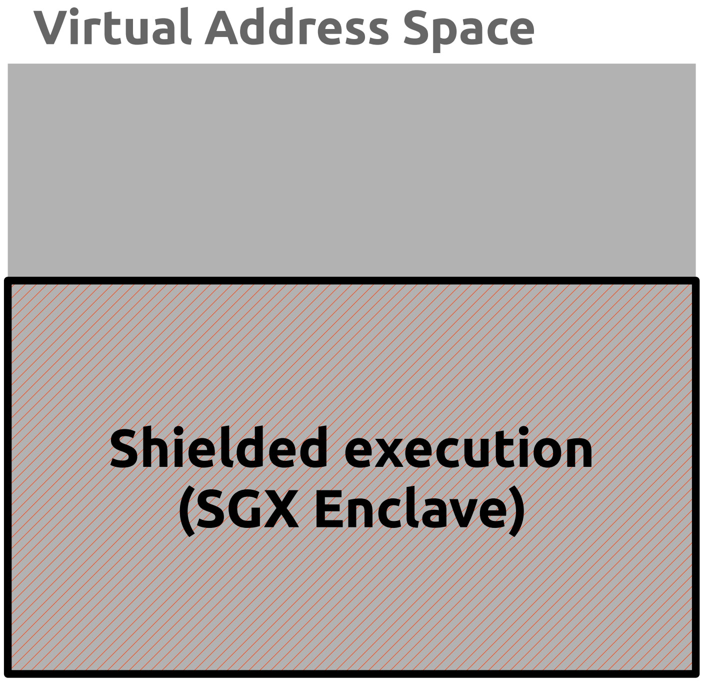
Security in the Cloud

- **Security** is a key barrier to adoption of cloud computing
- Attackers compromise **confidentiality** and **integrity**
  - ➡ **Malicious host (e.g., cloud provider)**
  - ➡ Software vulnerabilities



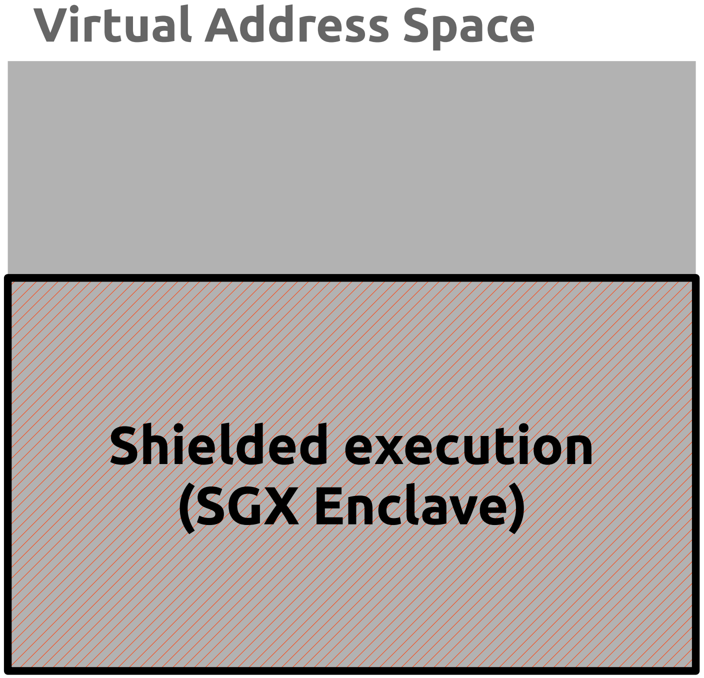
Security in the Cloud

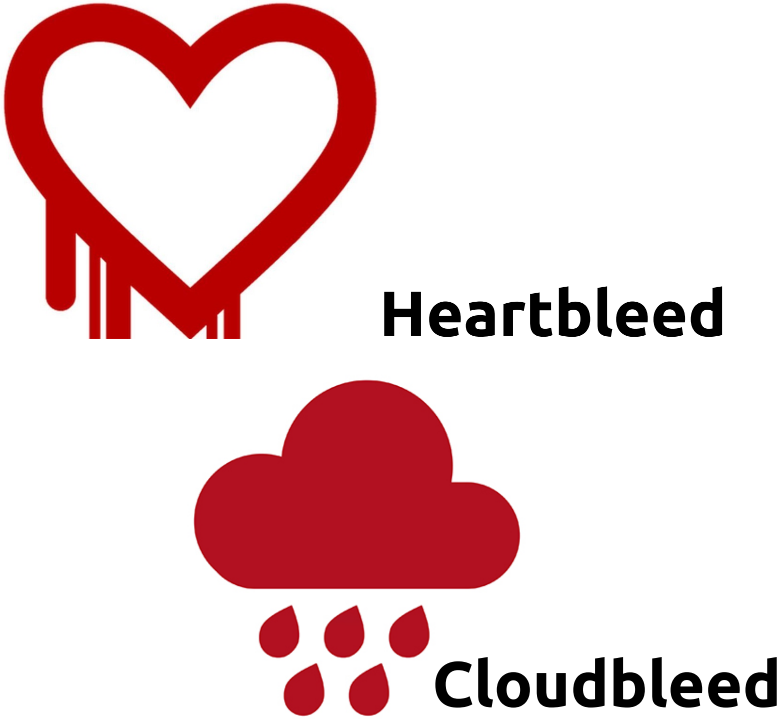
- **Security** is a key barrier to adoption of cloud computing
- Attackers compromise **confidentiality** and **integrity**
  - ➥ Malicious host (e.g., cloud provider)
  - ➥ **Software vulnerabilities**



Security in the Cloud

- **Security** is a key barrier to adoption of cloud computing
- Attackers compromise **confidentiality** and **integrity**
  - ➥ Malicious host (e.g., cloud provider)
  - ➥ **Software vulnerabilities**





Protecting against Attacks

SGX Enclave

(malicious host)

Protecting against Attacks

SGX Enclave

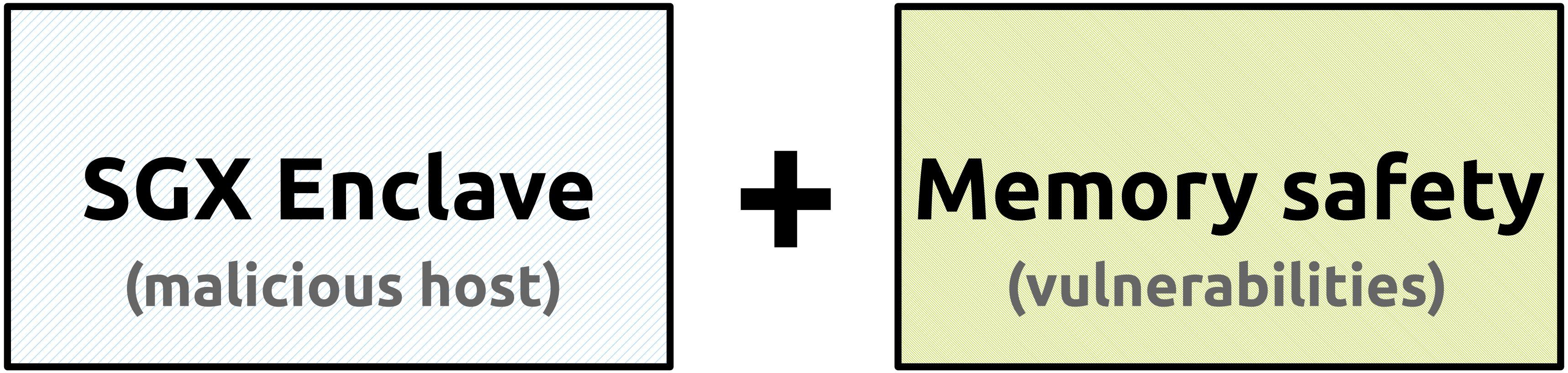
(malicious host)

+

Memory safety

(vulnerabilities)

Protecting against Attacks



SGX Enclave

(malicious host)

+

Memory safety

(vulnerabilities)

AddressSanitizer
(software-based)

Intel MPX
(hardware-based)

Protecting against Attacks

SGX Enclave

(malicious host)

+

Memory safety

(vulnerabilities)

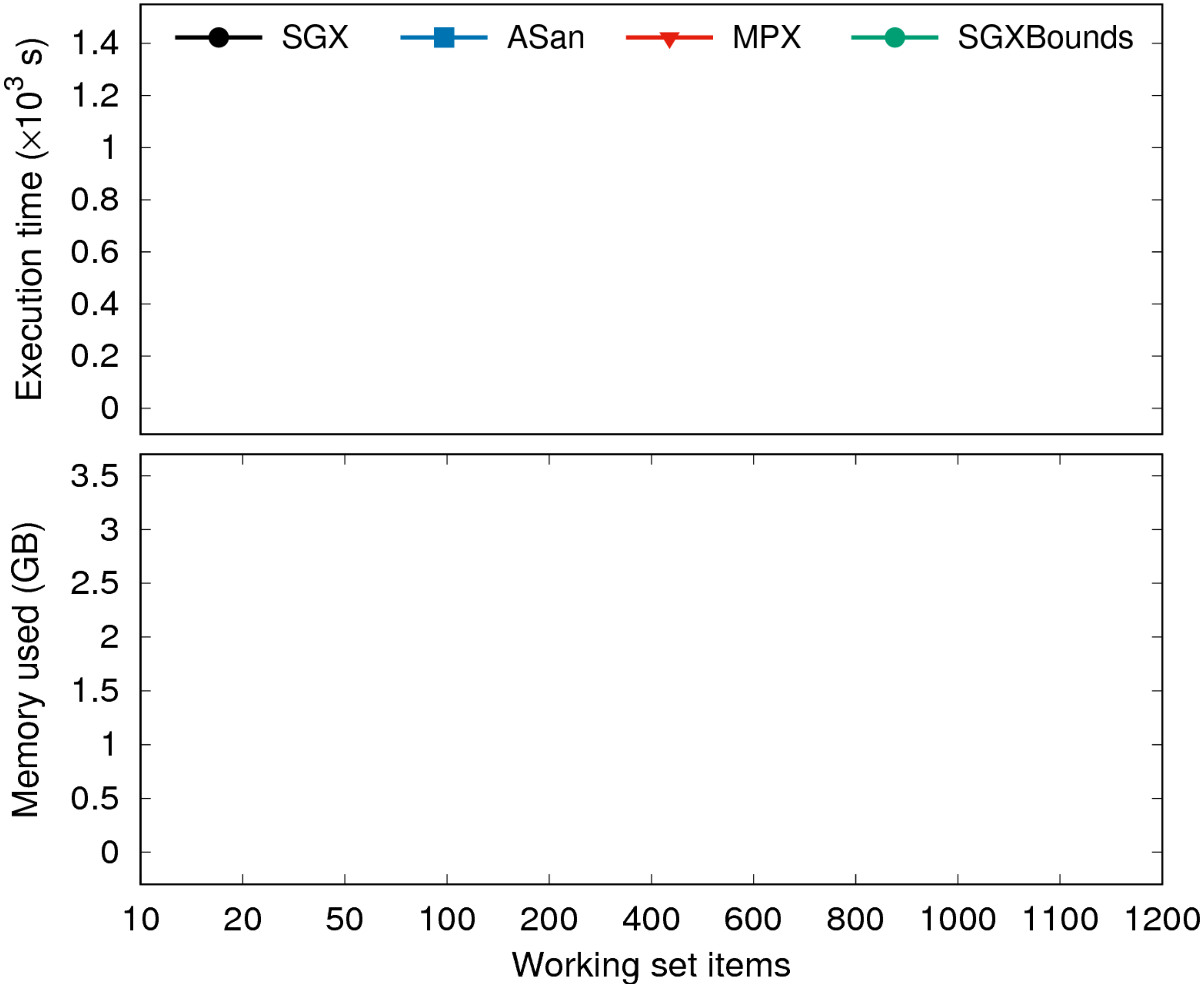
AddressSanitizer
(software-based)

Intel MPX
(hardware-based)

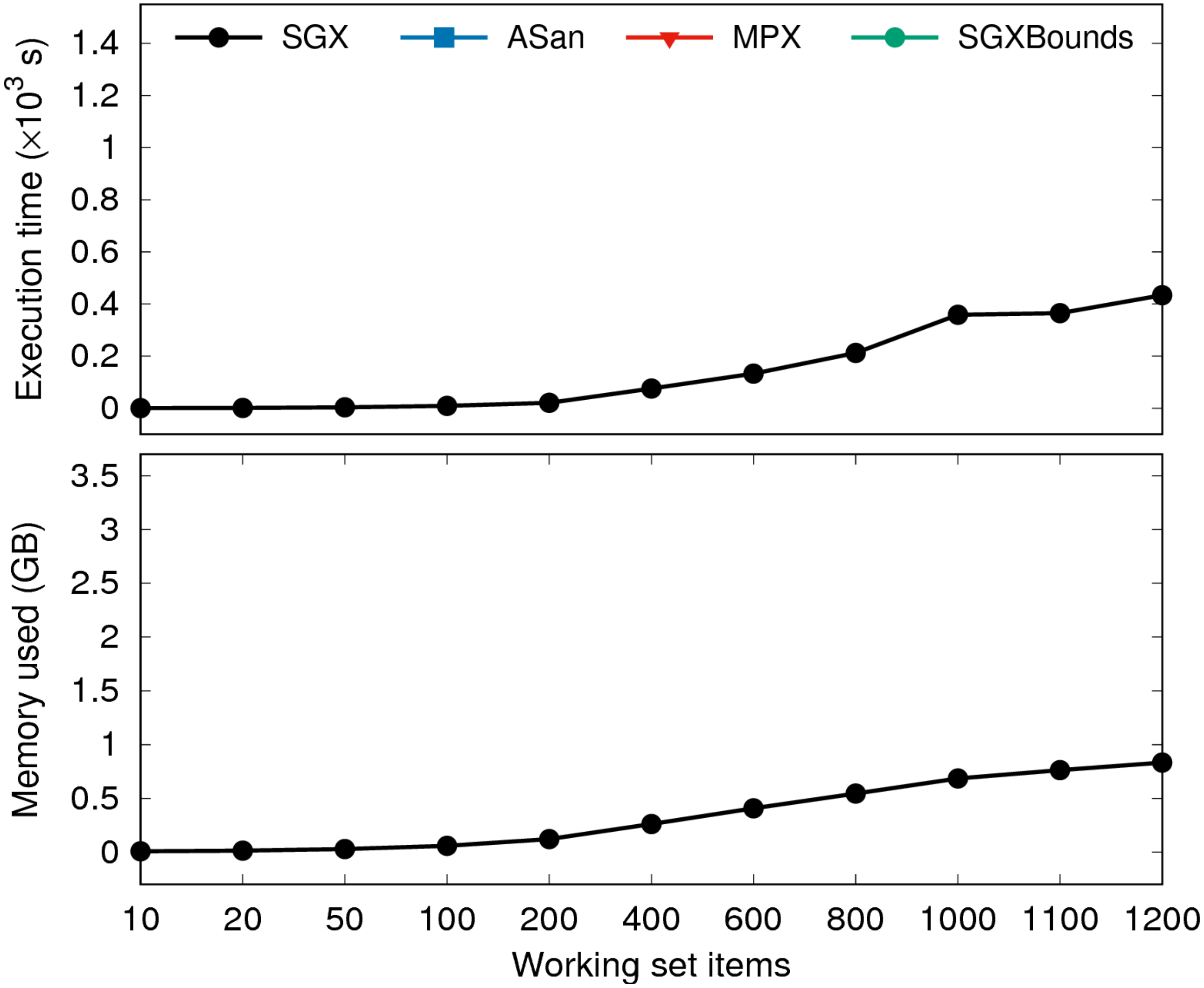
State-of-the-art memory-safety mechanisms are inefficient!

State-of-the-Art: SQLite example

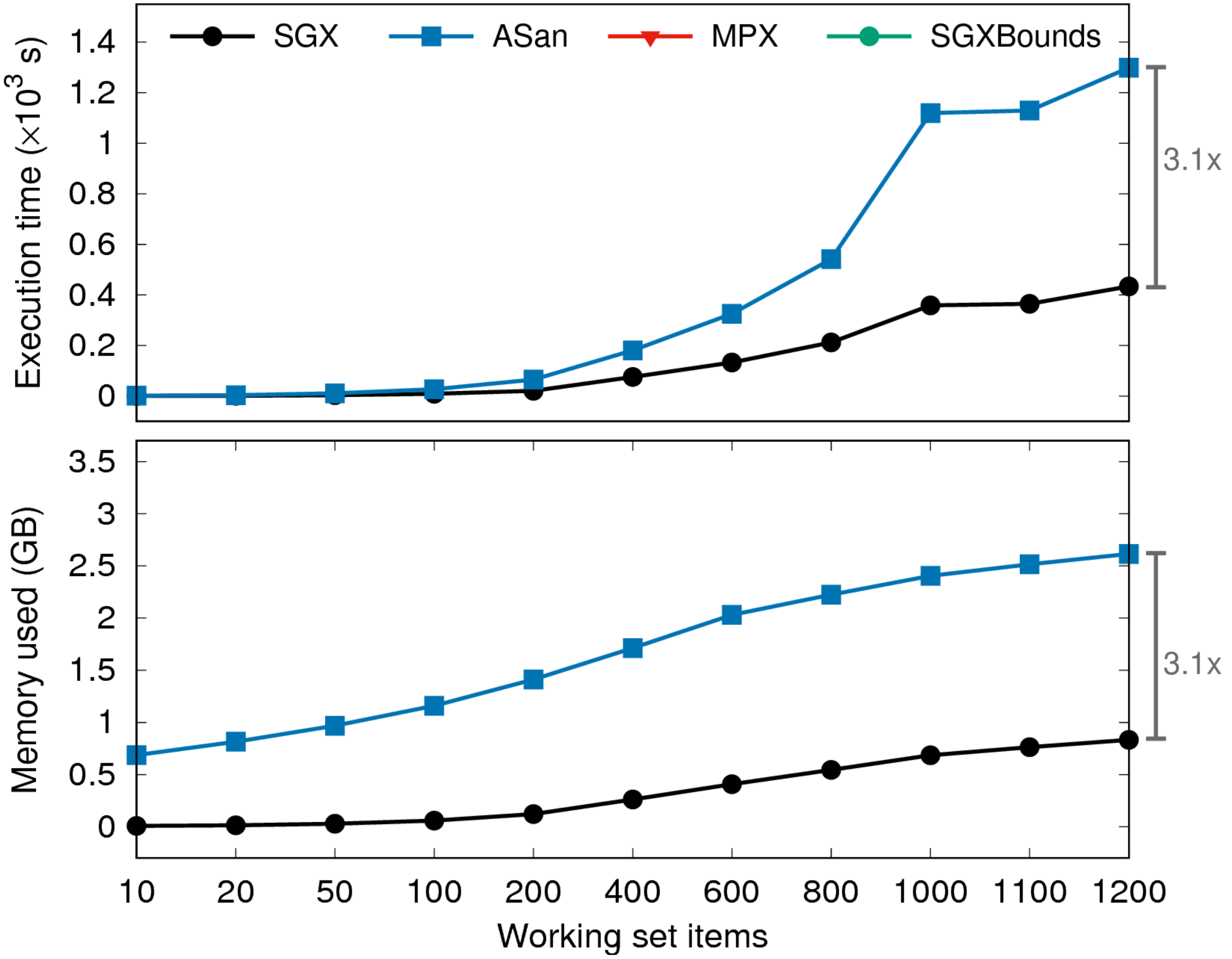
State-of-the-Art: SQLite example



State-of-the-Art: SQLite example

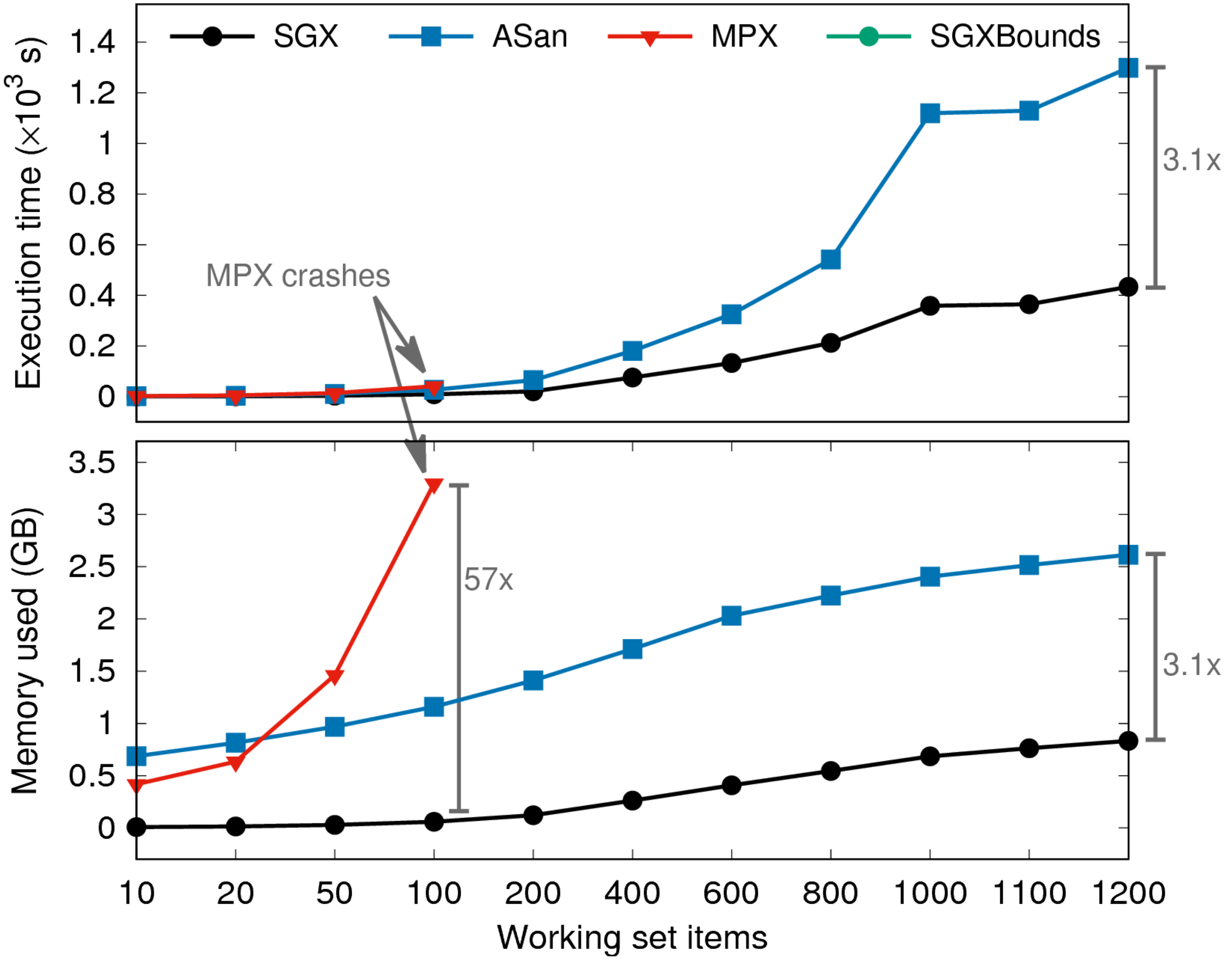


State-of-the-Art: SQLite example



lower better

State-of-the-Art: SQLite example



lower
better

SGX Enclave

(malicious host)

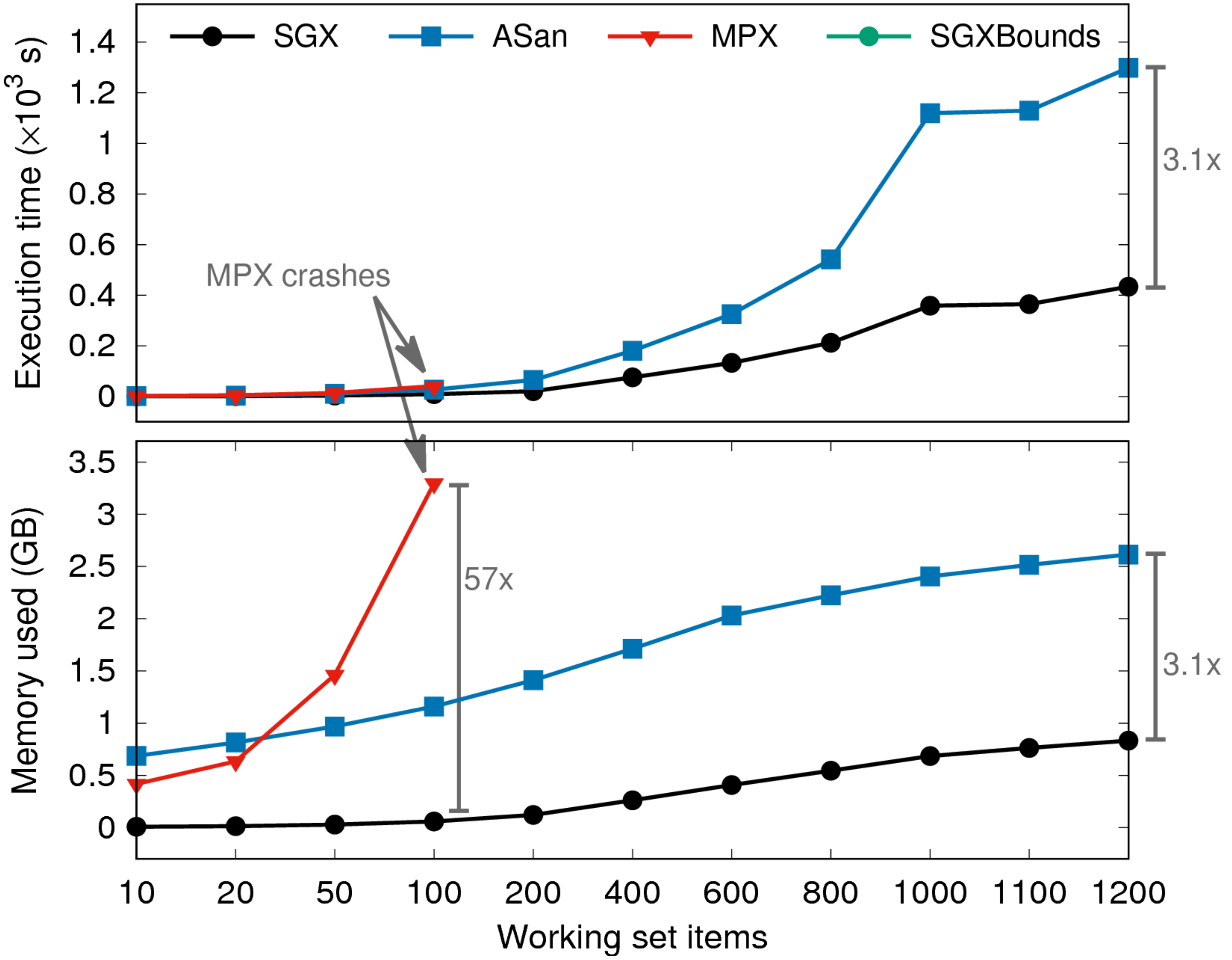
+

Memory safety

(vulnerabilities)

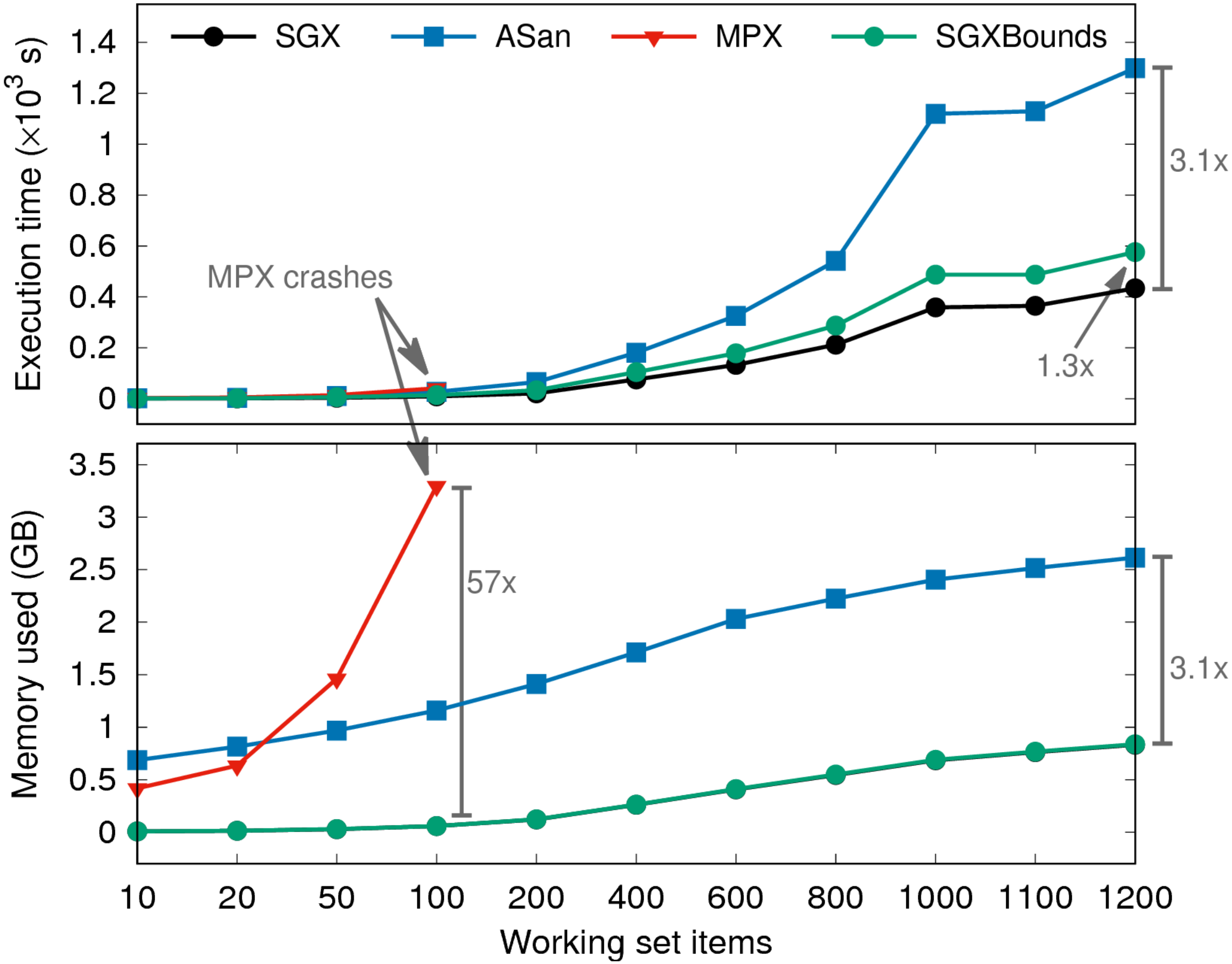
How to make it efficient?

State-of-the-Art: SQLite example



lower better

State-of-the-Art: SQLite example



SGXBounds is practical

– ~~Motivation~~

– Constraints of SGX enclaves

– Design of SGXBounds

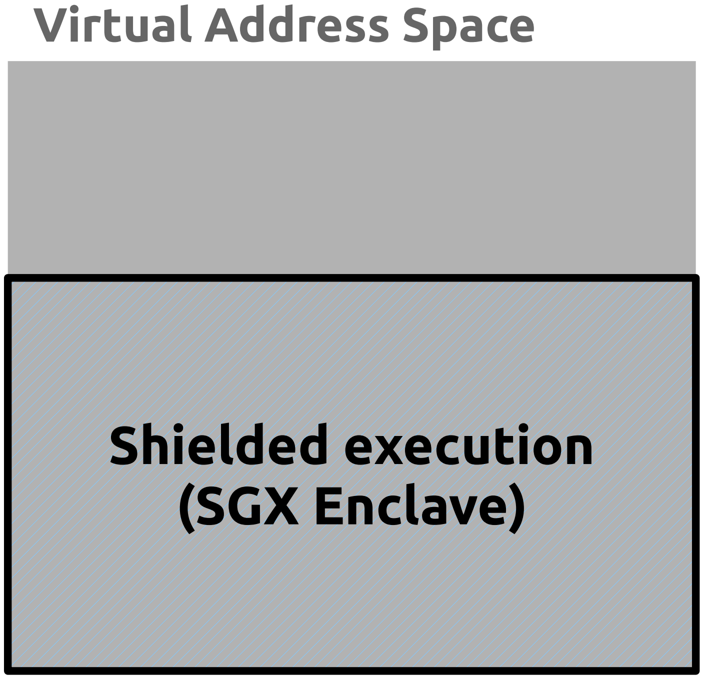
– Implementation of SGXBounds

– Evaluation

- ~~Motivation~~
- **Constraints of SGX enclaves**
- Design of SGXBounds
- Implementation of SGXBounds
- Evaluation

Constraints of SGX Enclaves

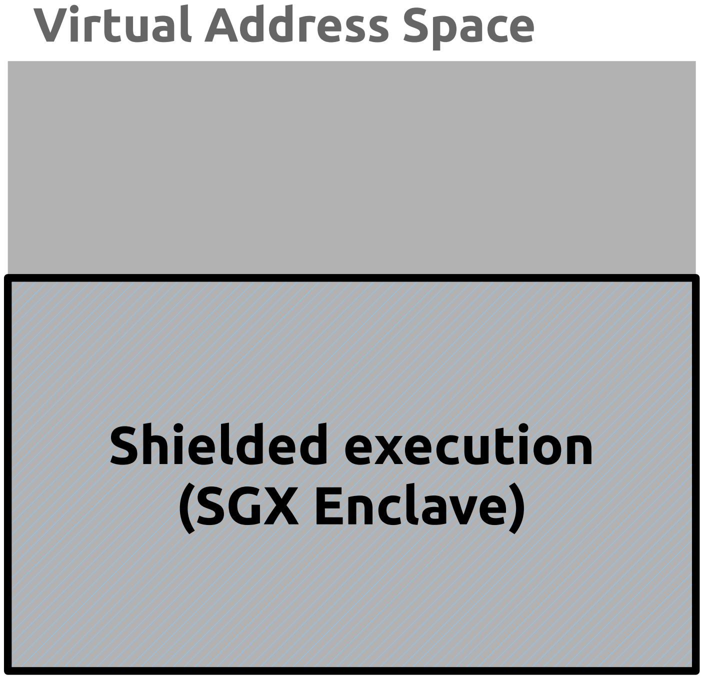
Why AddressSanitizer and Intel MPX perform poorly under SGX?



Constraints of SGX Enclaves

Why **AddressSanitizer** and Intel MPX perform poorly under SGX?

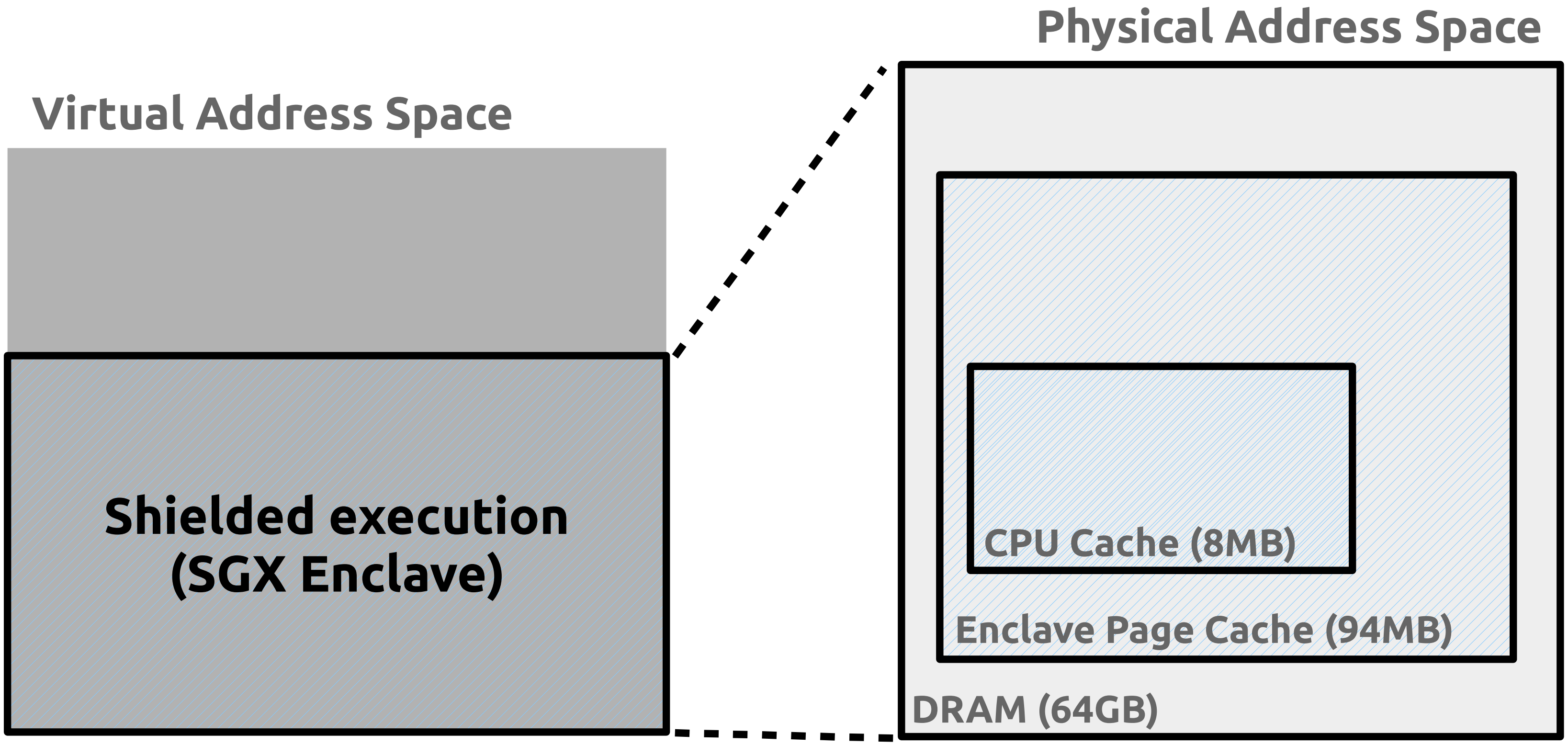
☹ **Increased latency** of memory accesses



Constraints of SGX Enclaves

Why **AddressSanitizer** and Intel MPX perform poorly under SGX?

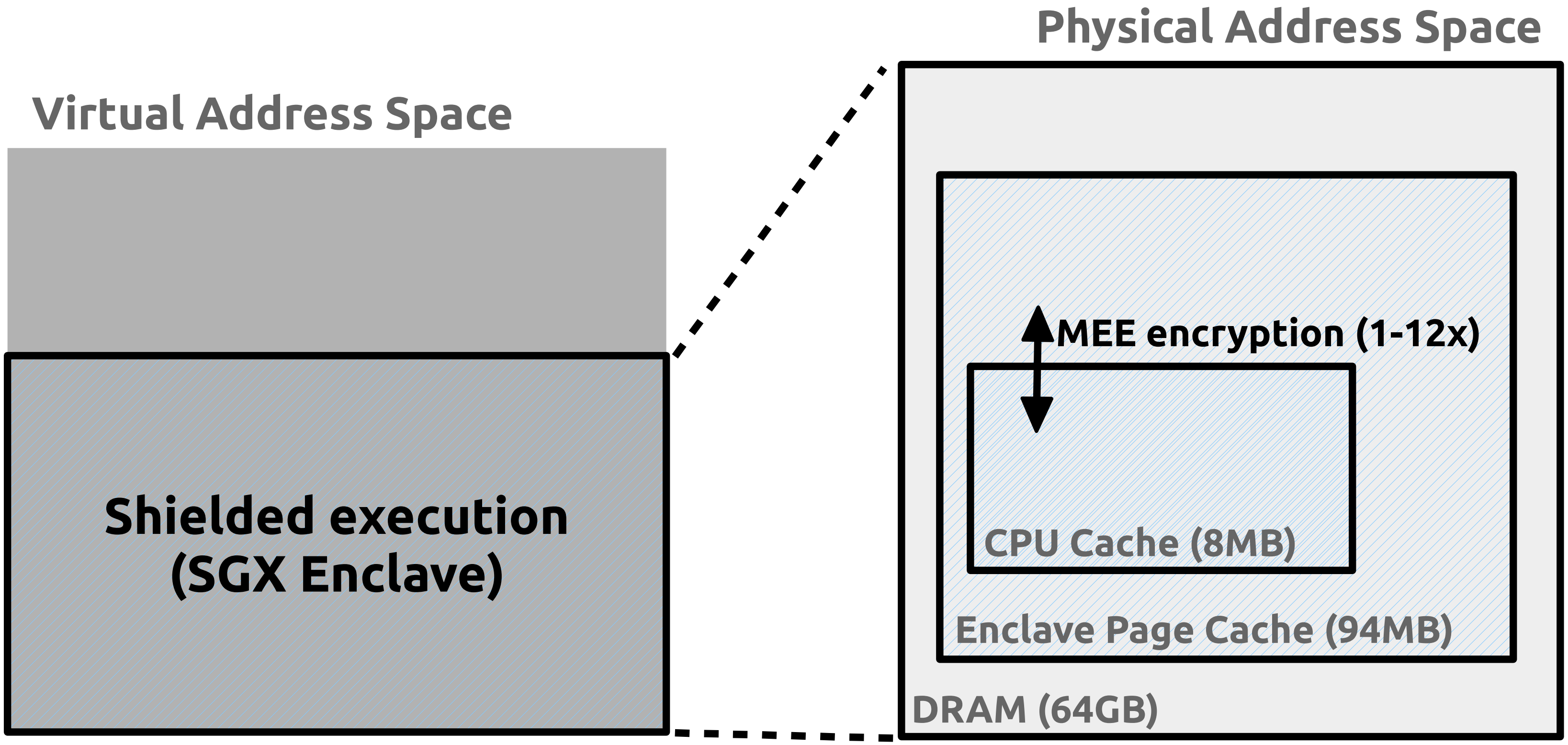
- ☹ **Increased latency** of memory accesses



Constraints of SGX Enclaves

Why **AddressSanitizer** and Intel MPX perform poorly under SGX?

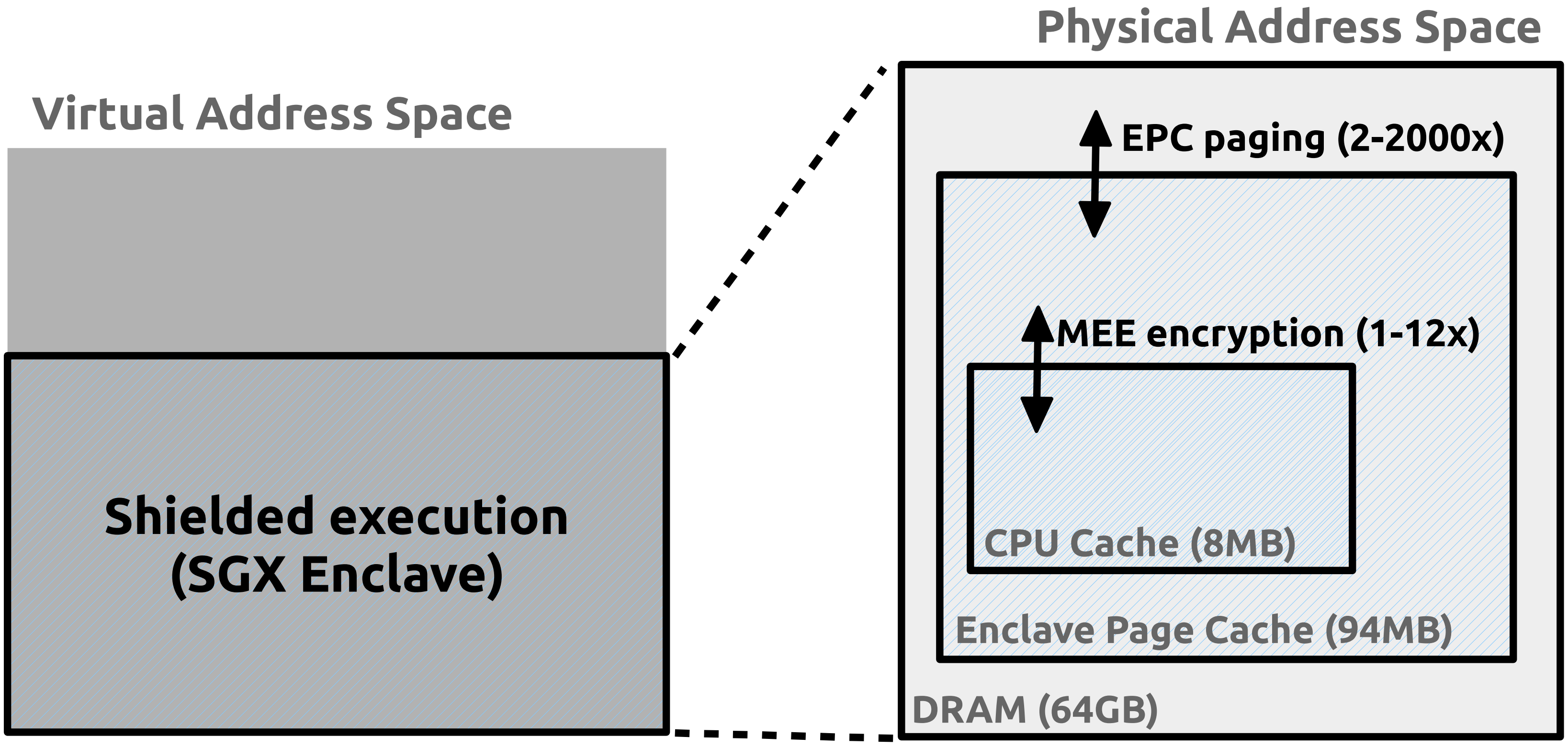
☹ **Increased latency** of memory accesses



Constraints of SGX Enclaves

Why **AddressSanitizer** and Intel MPX perform poorly under SGX?

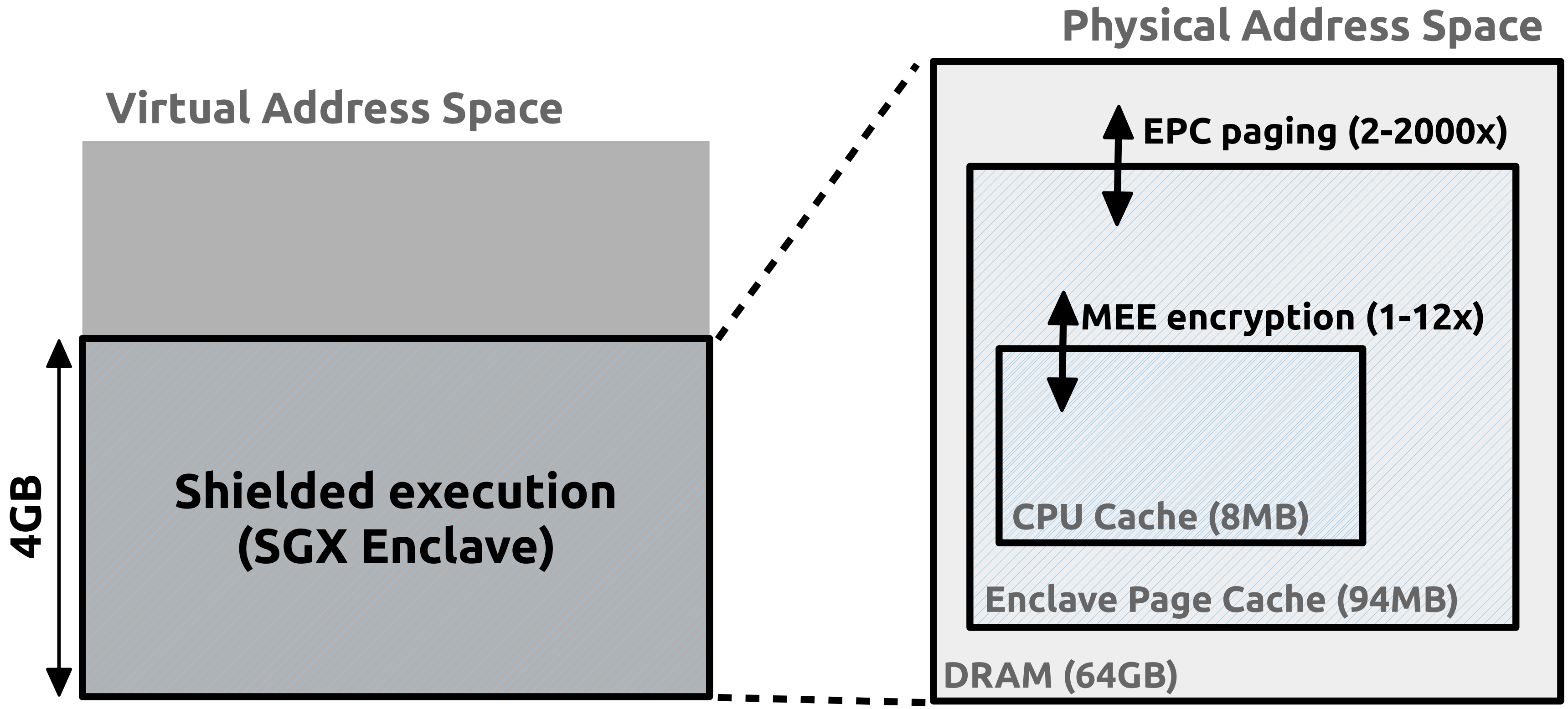
☹ **Increased latency** of memory accesses



Constraints of SGX Enclaves

Why AddressSanitizer and **Intel MPX** perform poorly under SGX?

- ☹ **Increased latency** of memory accesses
- ☹ **Limited** enclave memory (4GB)

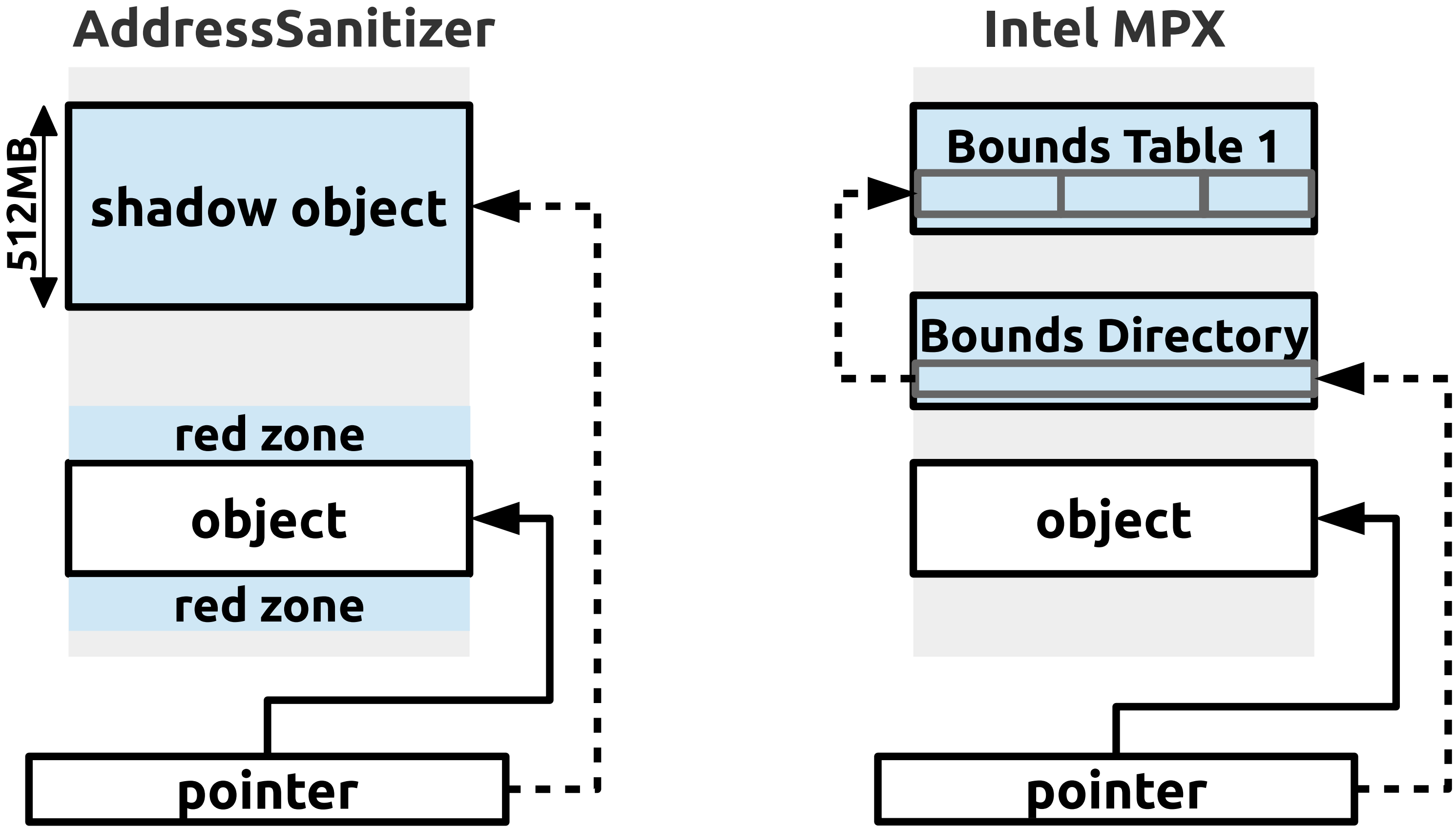


State-of-the-Art: Metadata Layout

Assumptions of AddressSanitizer and Intel MPX **violated** in SGX!

State-of-the-Art: Metadata Layout

Assumptions of AddressSanitizer and Intel MPX **violated** in SGX!

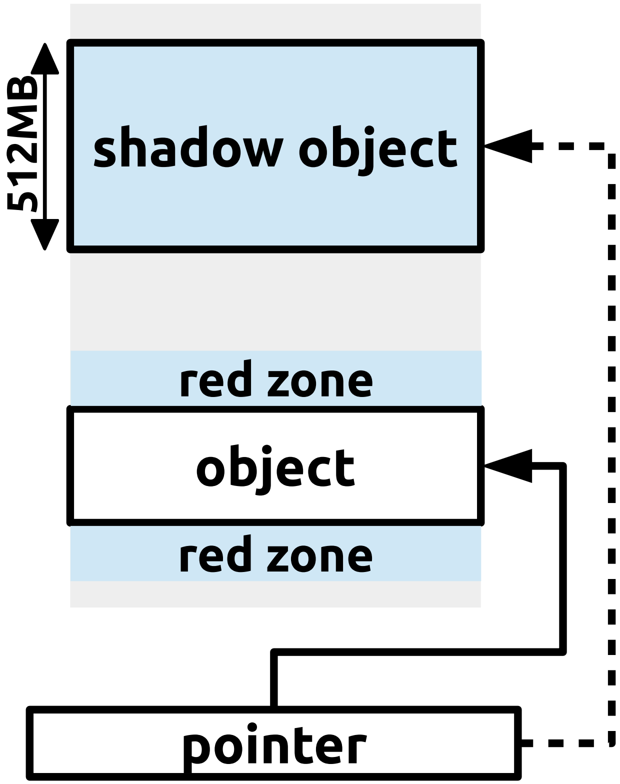


State-of-the-Art: Metadata Layout

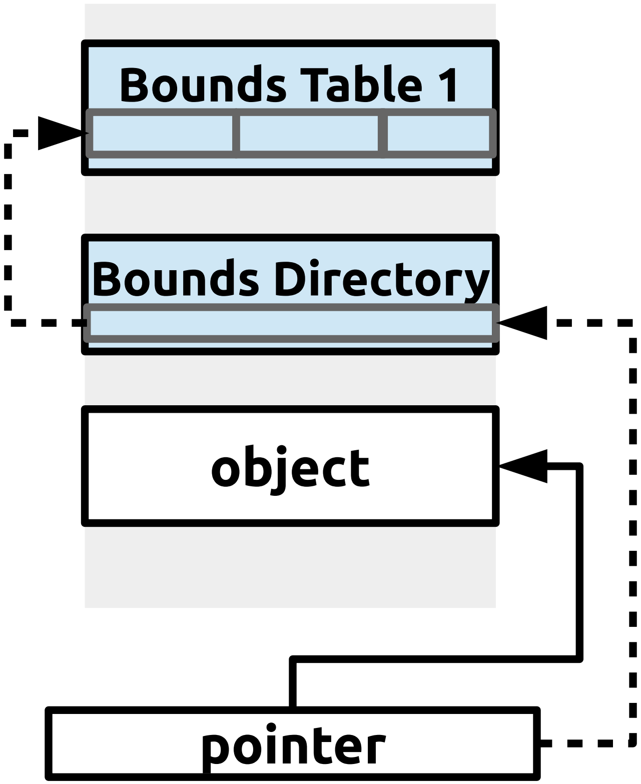
Assumptions of AddressSanitizer and Intel MPX **violated** in SGX!

- ☹ Fast accesses to metadata
- ☹ Almost endless memory

AddressSanitizer



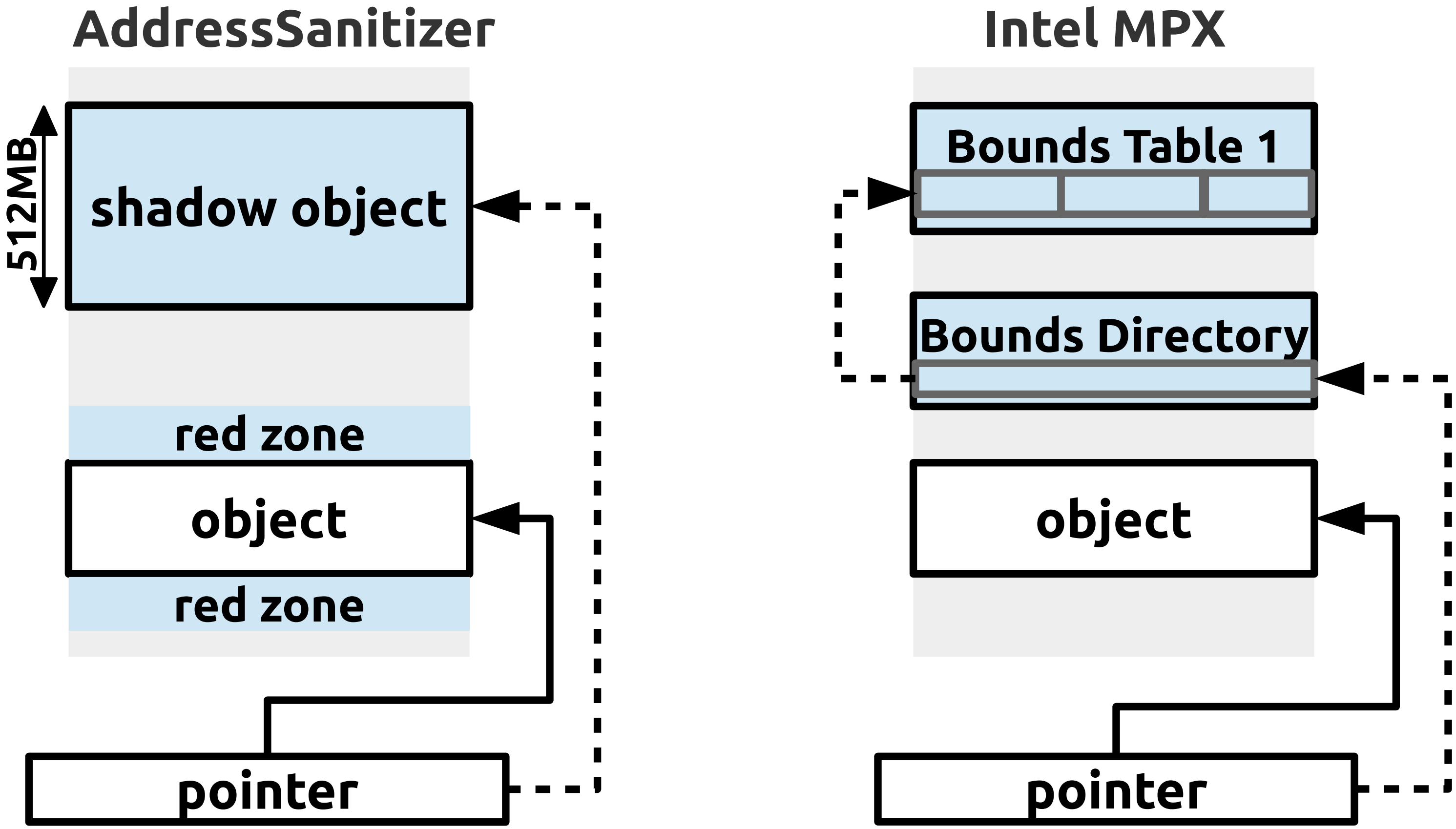
Intel MPX



State-of-the-Art: Metadata Layout

Assumptions of AddressSanitizer and Intel MPX **violated** in SGX!

- ☹ Fast accesses to metadata ≠ increased latency
- ☹ Almost endless memory ≠ limited enclave memory

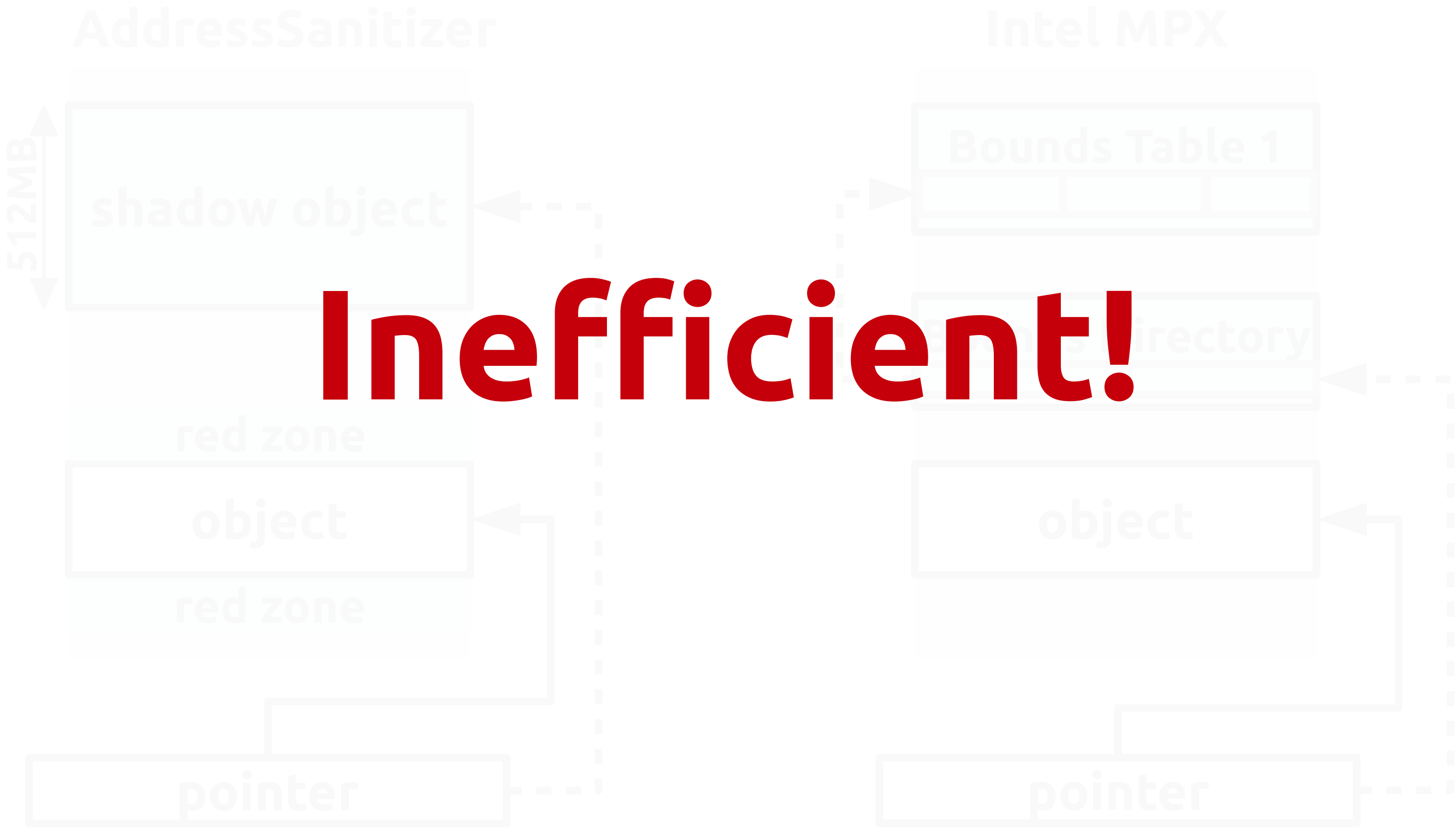


State-of-the-Art: Metadata Layout

Assumptions of AddressSanitizer and Intel MPX **violated** in SGX!

☹ Fast accesses to metadata ≠ increased latency

☹ Almost endless memory ≠ limited enclave memory



Inefficient!

- ~~Motivation~~
- ~~Constraints of SGX enclaves~~
- **Design of SGXBounds**
- Implementation of SGXBounds
- Evaluation

SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

- ☺ Increased latency → **minimize accesses** to metadata

SGXBounds: Metadata Layout

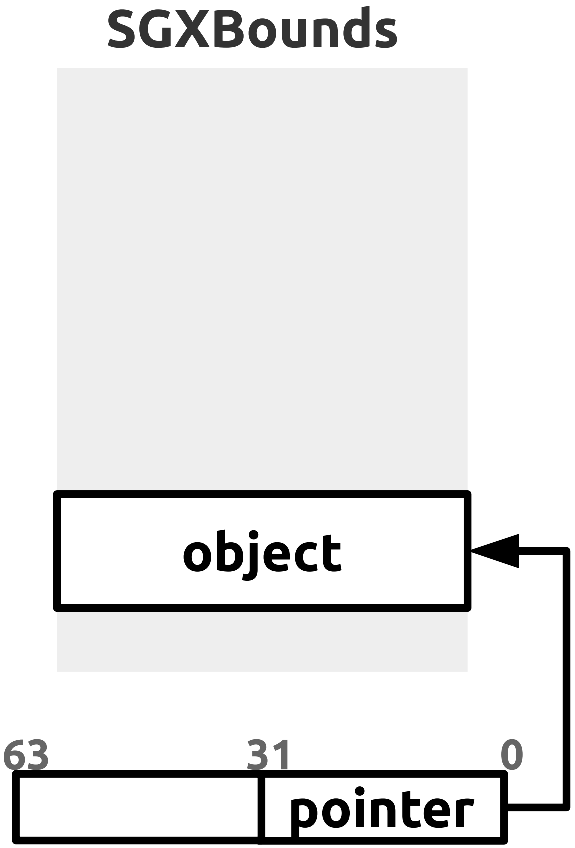
Memory contraints of **SGX** dictated design of **SGXBounds**

- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata

SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

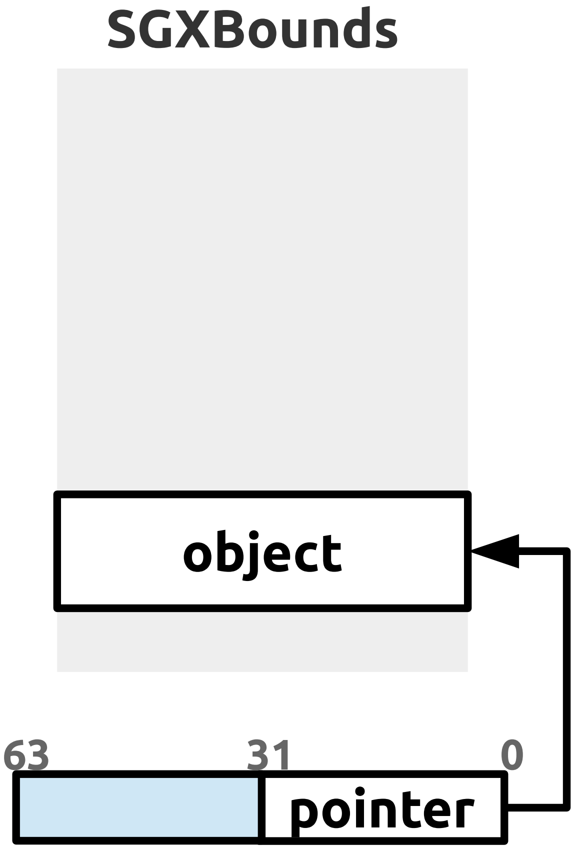
- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata



SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

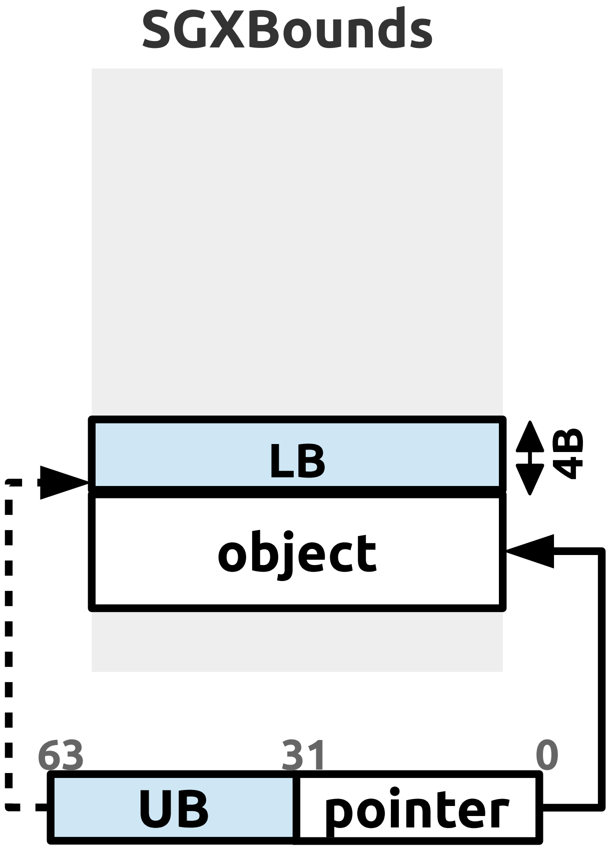
- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata



SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

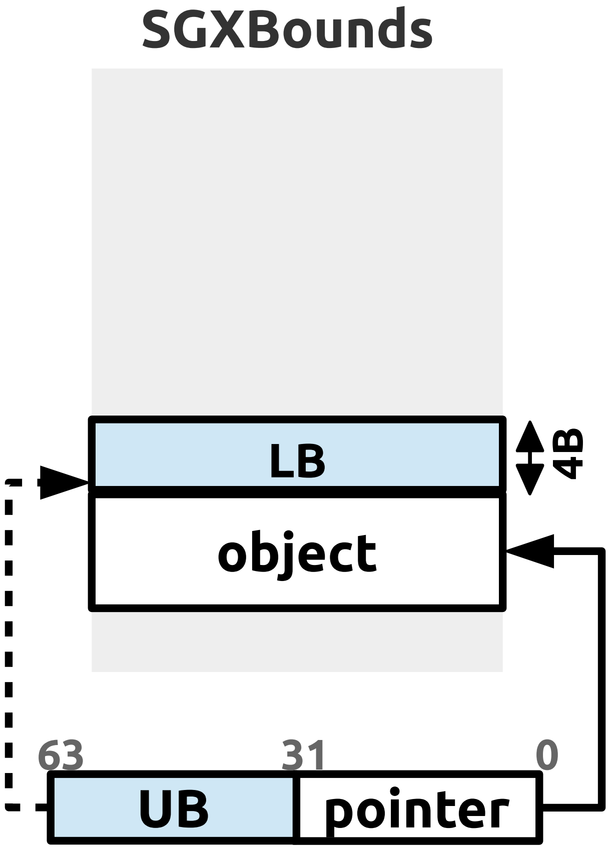
- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata



SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata



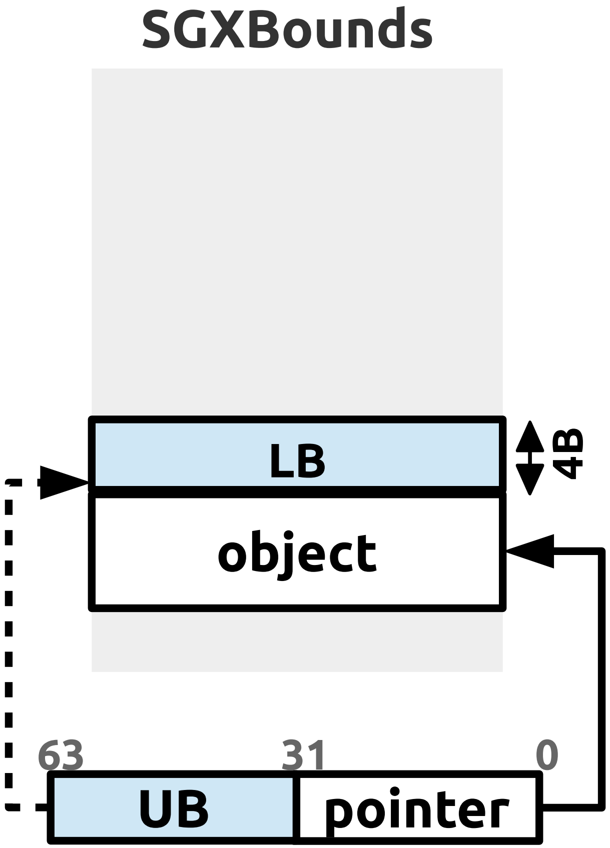
– **Upper bound (UB)** in pointer

– **Lower bound (LB)** per object

SGXBounds: Metadata Layout

Memory contraints of **SGX** dictated design of **SGXBounds**

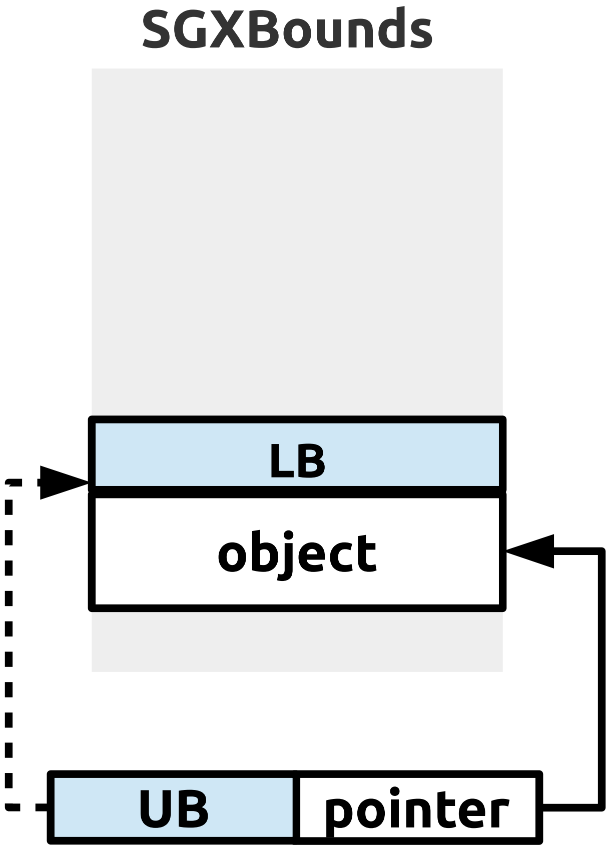
- ☺ Increased latency → **minimize accesses** to metadata
- ☺ Limited enclave memory → **minimize space** of metadata



- **Upper bound (UB)** in pointer
- **Lower bound (LB)** per object
- Out-of-the-box **multithreading** (unlike MPX)

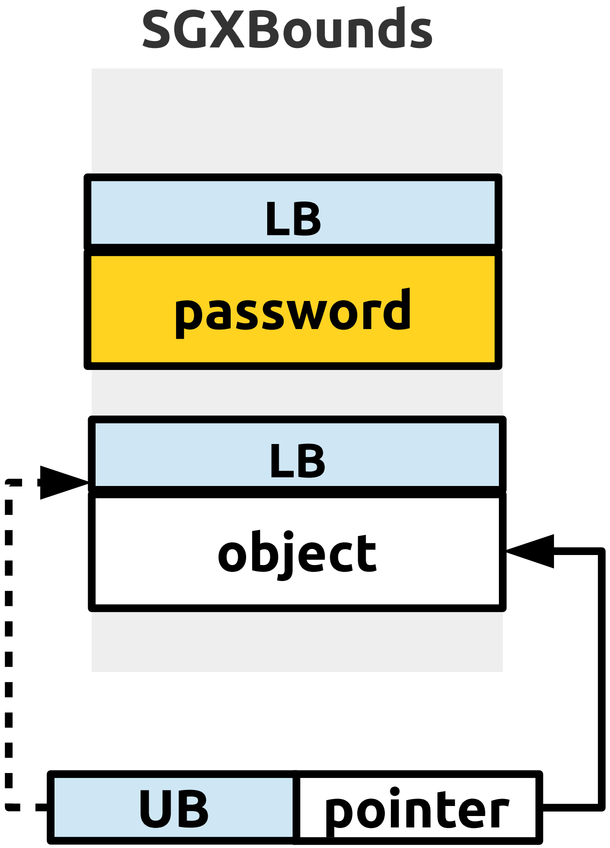
SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?



SGXBounds: Detecting Vulnerabilities

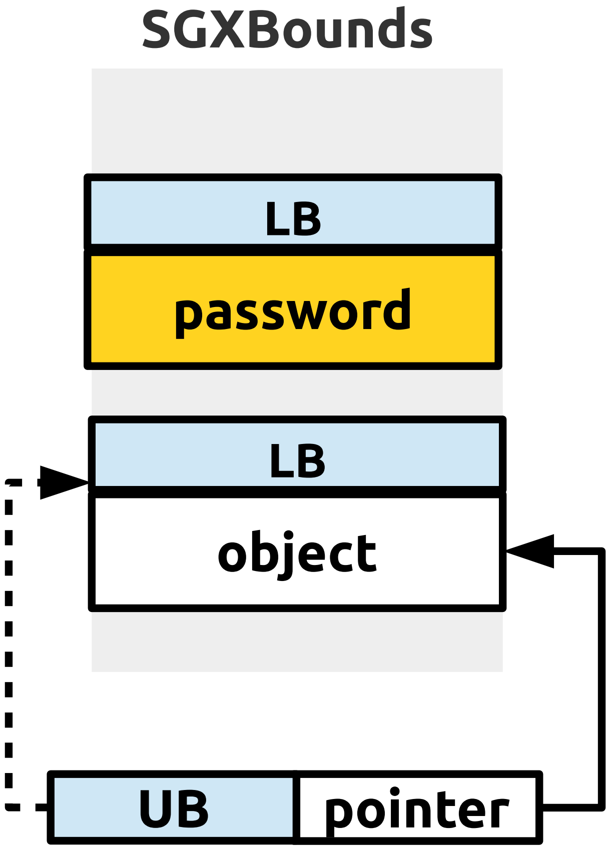
How SGXBounds **detects vulnerabilities** like Heartbleed?



SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?

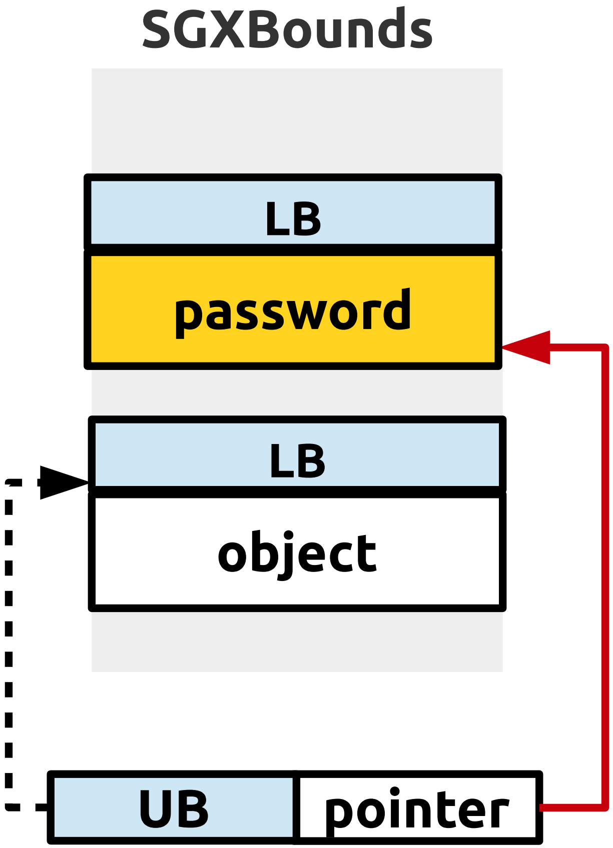
- ☹ Data leak through `write(socket, pointer, objlen)`



SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?

☹ Data leak through `write(socket, pointer, objlen)`

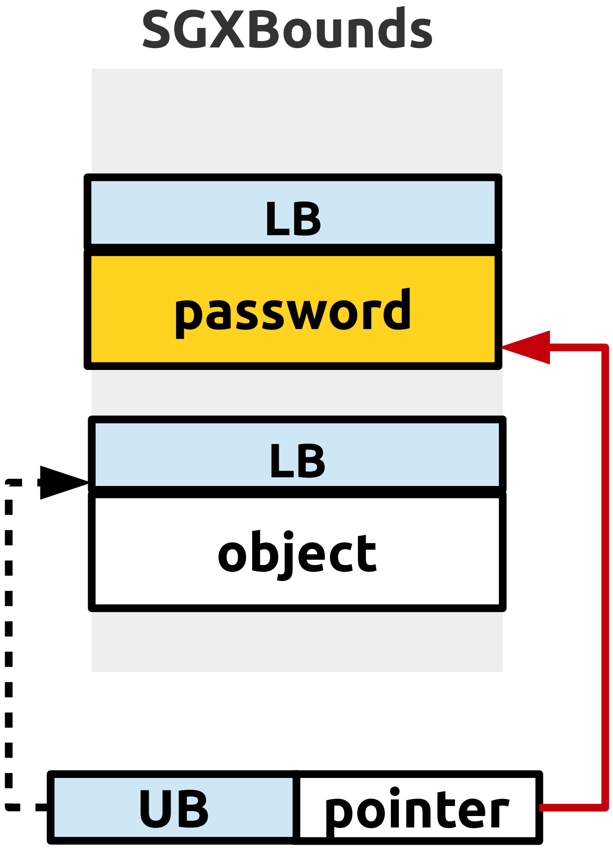


SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?

☹ Data leak through **`write`**`(socket, `**`pointer`**`, objlen)`

☺ Protect using efficient **bounds checks**



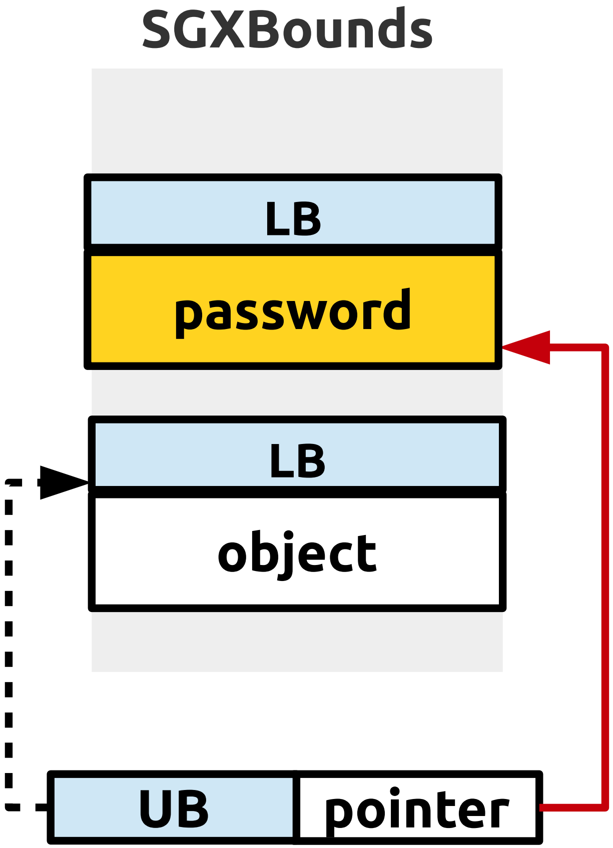
Bounds-check before each memory access:

`LB ≤ `**`pointer`**` ≤ UB`

SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?

- ☹ Data leak through **`write`**`(socket, `**`pointer`**`, objlen)`
- ☺ Protect using efficient **bounds checks**



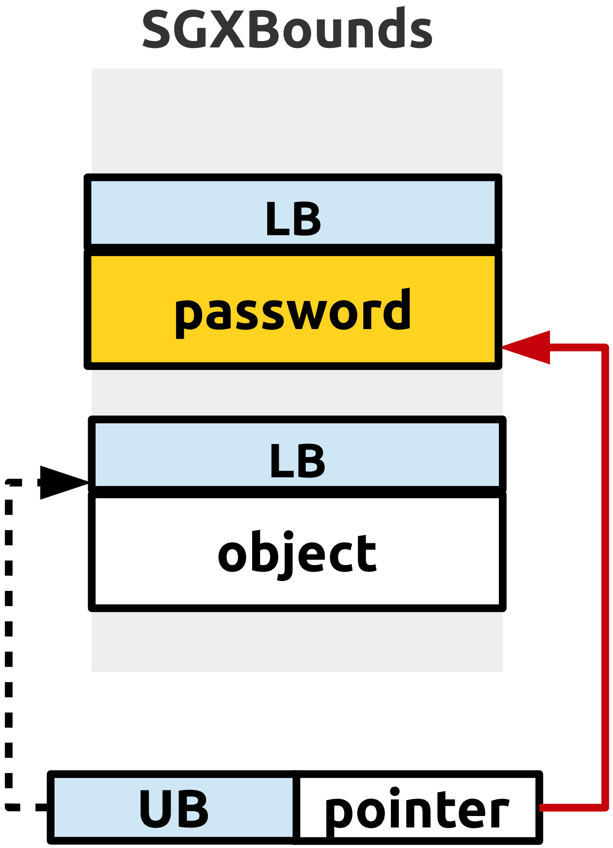
Bounds-check before each memory access:

$$\underbrace{LB \leq \textbf{pointer} \leq UB}_{\text{embedded in tagged pointer}}$$

SGXBounds: Detecting Vulnerabilities

How SGXBounds **detects vulnerabilities** like Heartbleed?

- ☹ Data leak through **`write`**`(socket, `**`pointer`**`, objlen)`
- ☺ Protect using efficient **bounds checks**



Bounds-check before each memory access:

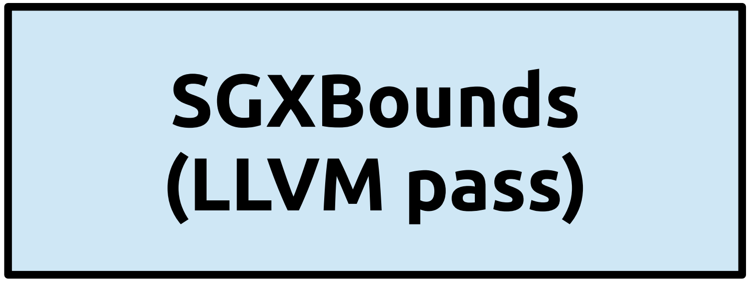
$$\underbrace{LB \le \textbf{pointer} \le UB}$$

embedded in tagged pointer

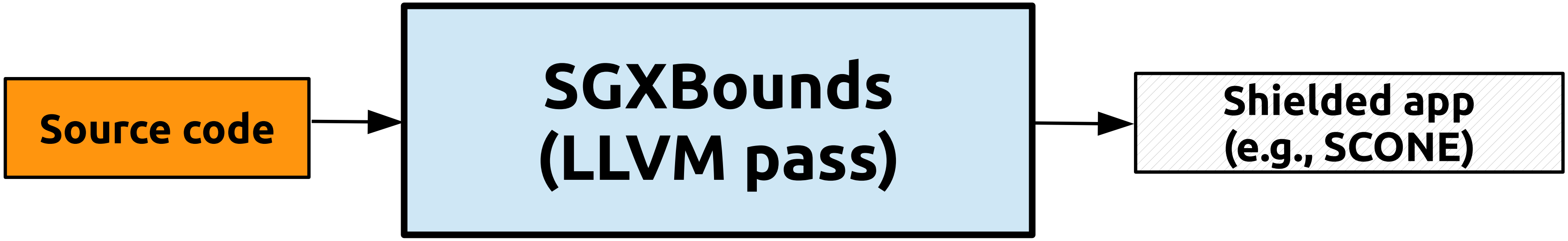
loaded from memory based on UB

- ~~Motivation~~
- ~~Constraints of SGX enclaves~~
- ~~Design of SGXBounds~~
- **Implementation of SGXBounds**
- Evaluation

SGXBounds: Implementation



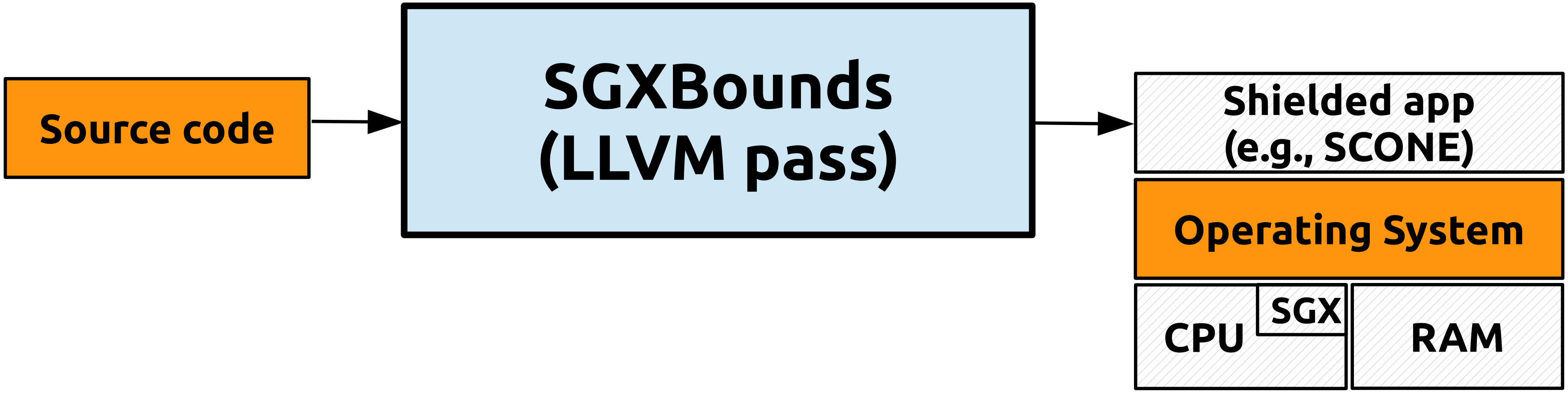
SGXBounds: Implementation


Source code

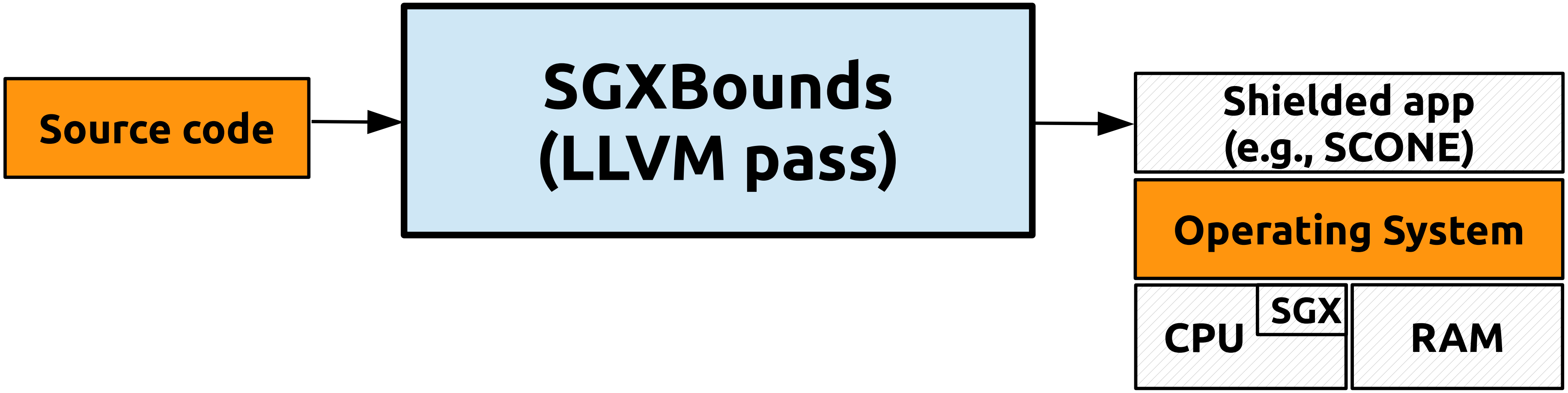
SGXBounds (LLVM pass)

Shielded app (e.g., SCONE)

SGXBounds: Implementation

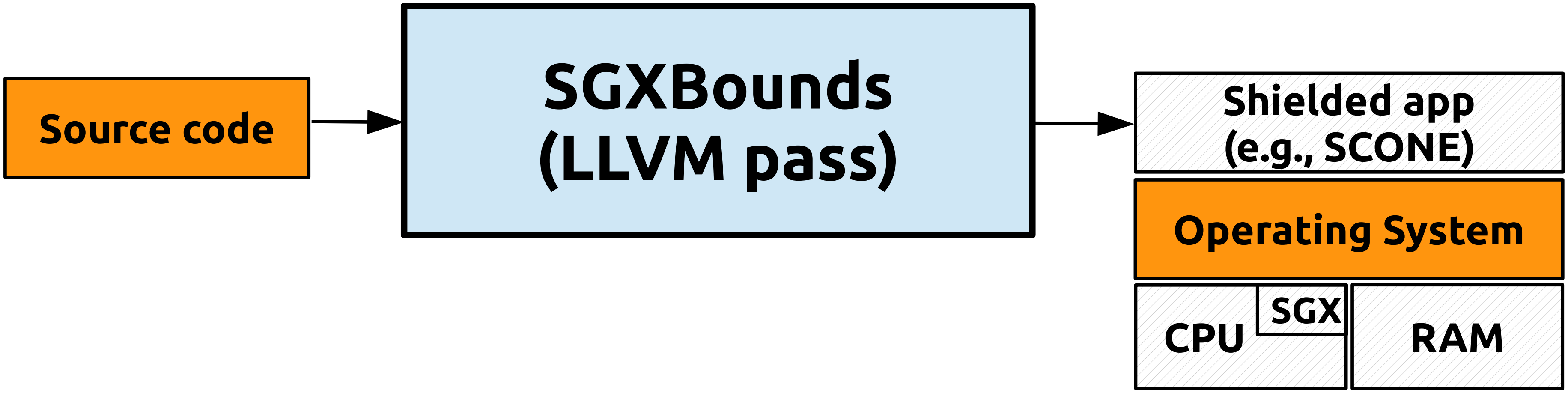


SGXBounds: Implementation



Advanced features:

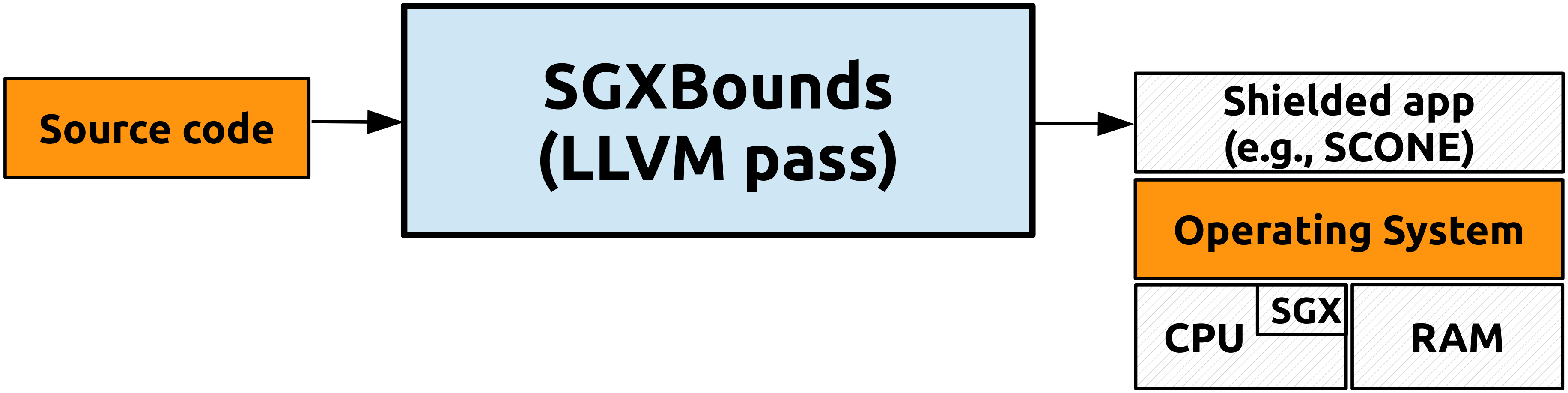
SGXBounds: Implementation



Advanced features:

- ➥ **Tolerating** errors with **boundless memory**
- ➥ **Metadata management** support
- ➥ Compile-time **optimizations**

SGXBounds: Implementation



Advanced features:

- ➥ **Tolerating** errors with **boundless memory**
- ➥ **Metadata management** support
- ➥ Compile-time **optimizations**

See paper for details

- ~~Motivation~~
- ~~Constraints of SGX enclaves~~
- ~~Design of SGXBounds~~
- ~~Implementation of SGXBounds~~
- **Evaluation**

- ~~Motivation~~
- ~~Constraints of SGX enclaves~~
- ~~Design of SGXBounds~~
- ~~Implementation of SGXBounds~~
- **Evaluation**
    - ➥ Benchmark suites
    - ➥ Case studies
    - ➥ Security

Benchmark Suites

	ASan	MPX	SGXBounds

Benchmark Suites

	ASan	MPX	SGXBounds
Phoenix	1.41	2.27	**1.13**

Benchmark Suites

	ASan	MPX	SGXBounds
Phoenix	1.41	2.27	**1.13**
PARSEC	1.60	1.43*	**1.20**

* some programs failed due to insufficient memory

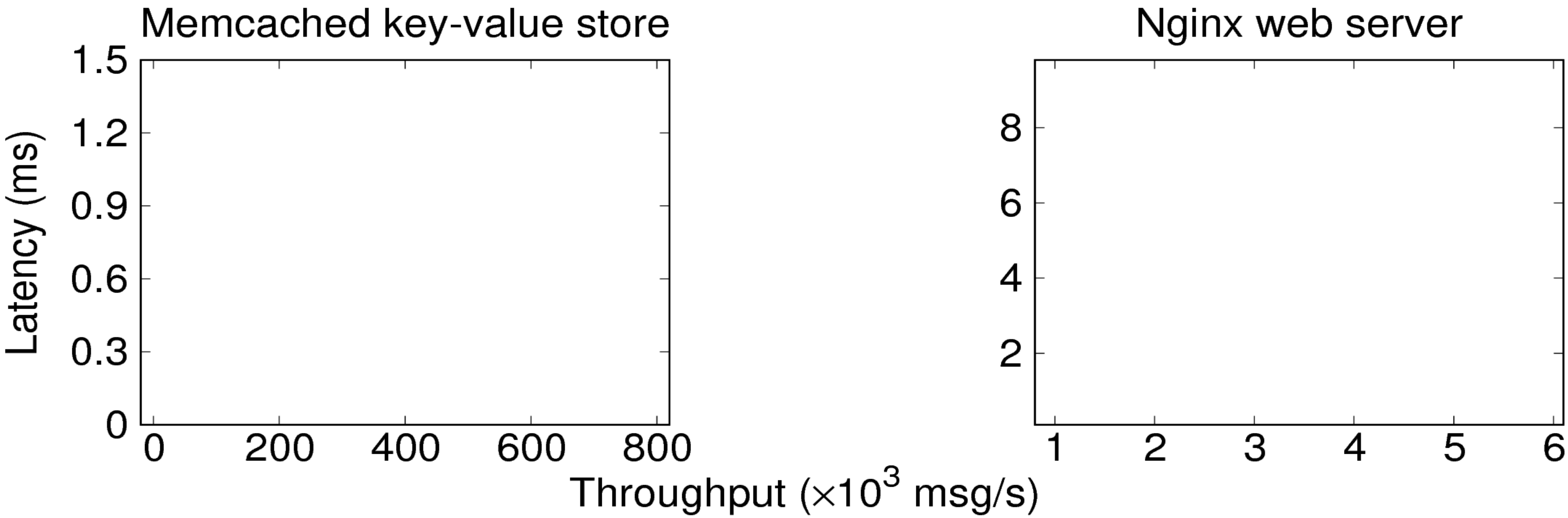
Benchmark Suites

	ASan	MPX	SGXBounds
Phoenix	1.41	2.27	**1.13**
PARSEC	1.60	1.43*	**1.20**
SPEC	1.76	1.52*	**1.41**

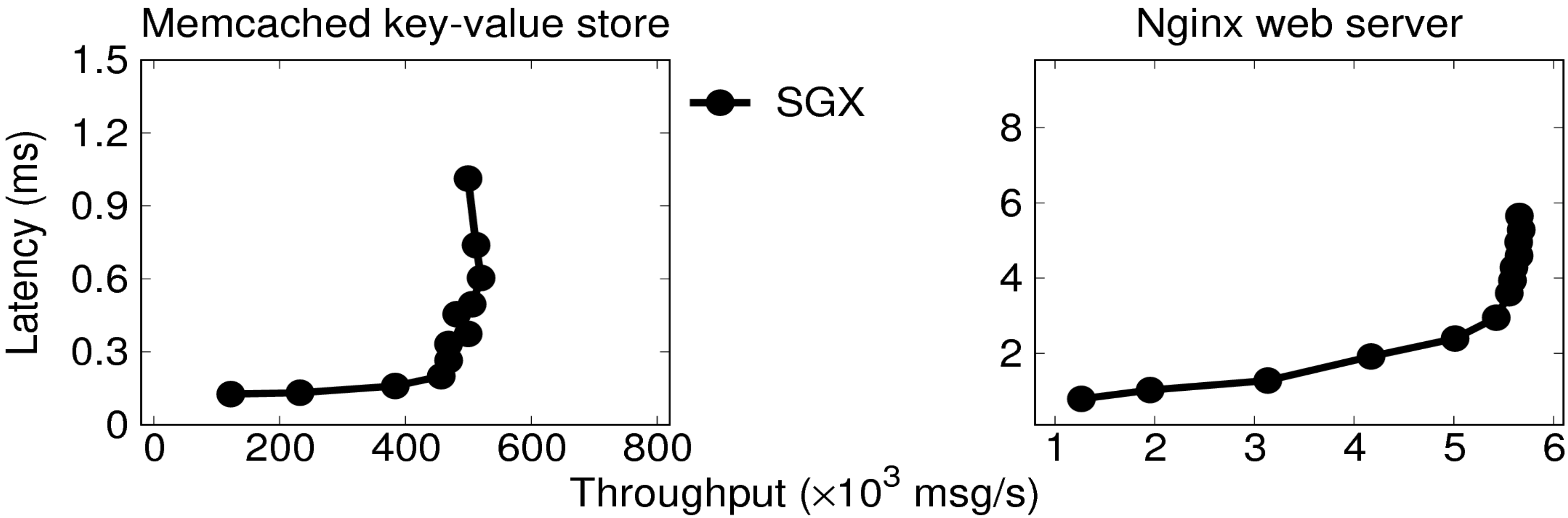
* some programs failed due to insufficient memory

Case Studies

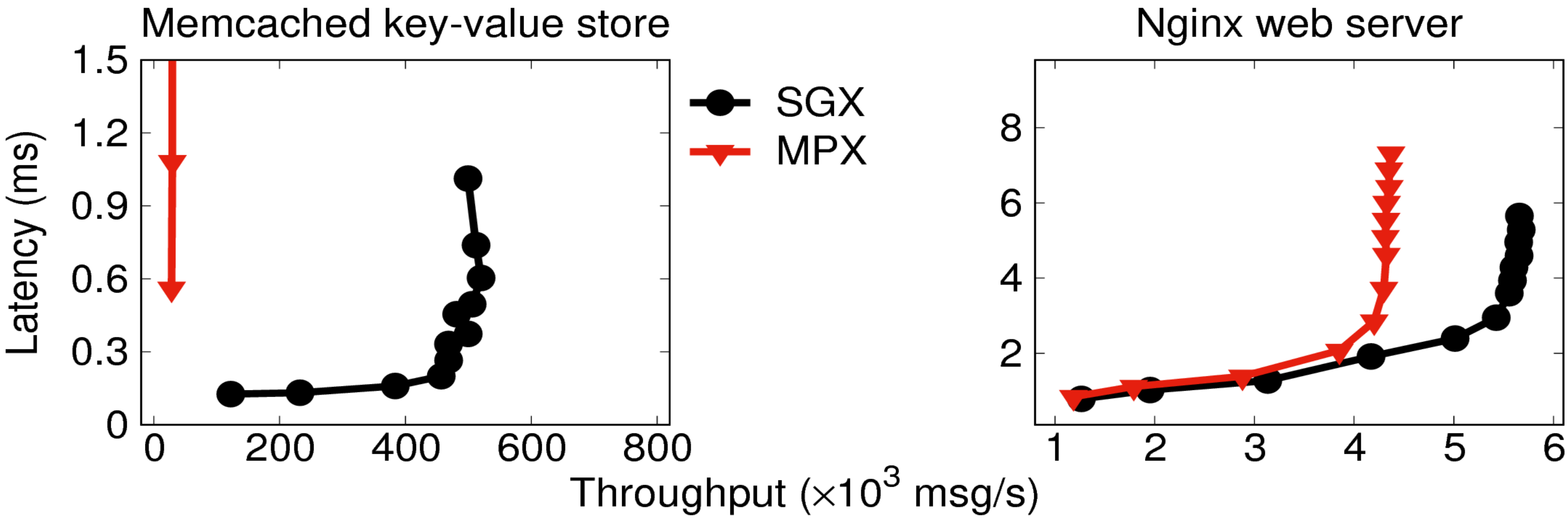
Case Studies



Case Studies

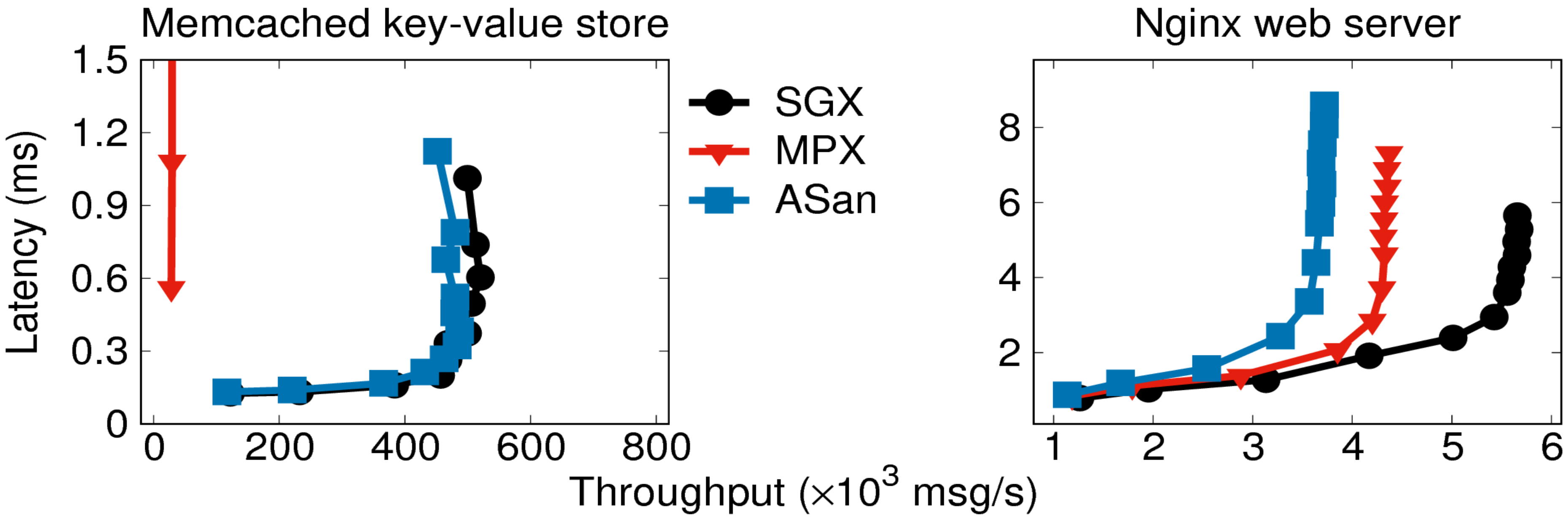


Case Studies



☹ **MPX**: EPC thrashing on Memcached

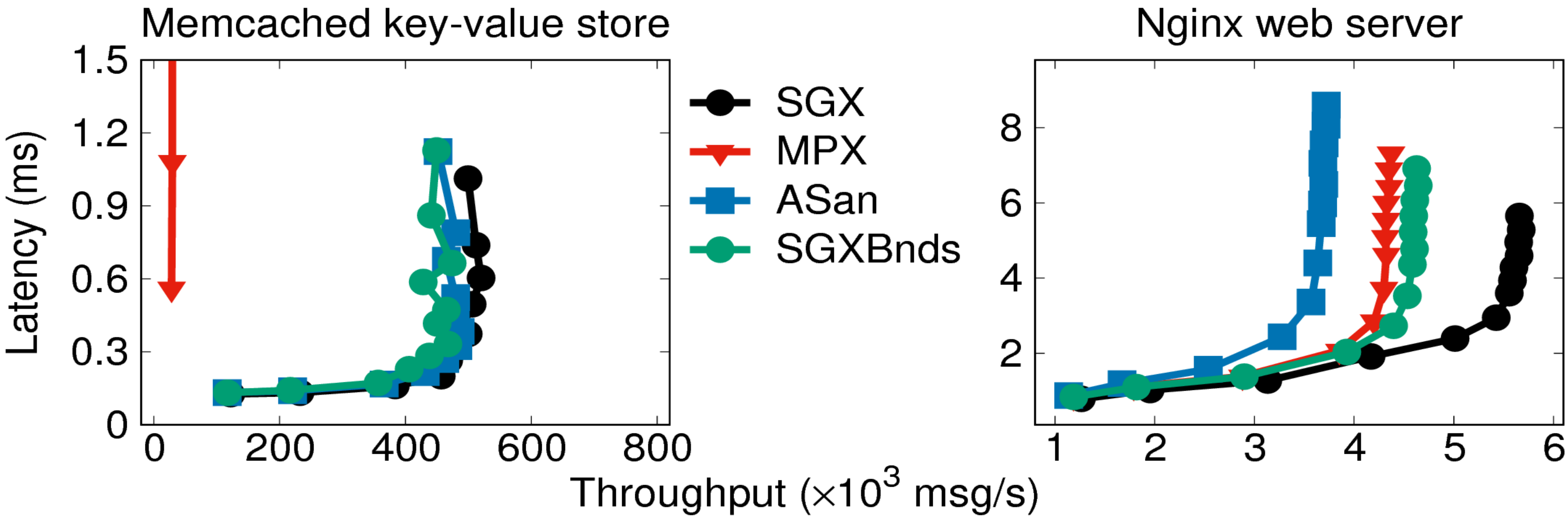
Case Studies



☹ **MPX**: EPC thrashing on Memcached

☹ **ASan**: metadata overload on Nginx

Case Studies



☹ **MPX**: EPC thrashing on Memcached

☹ **ASan**: metadata overload on Nginx

☺ **SGXBounds**: no corner cases

Security guarantees

Security guarantees

- ☺ **RIPE** synthetic benchmark:
  - ➥ **Similar guarantees** as ASan and MPX

Security guarantees

- ☺ **RIPE** synthetic benchmark:
  - ➥ **Similar guarantees** as ASan and MPX
- ☺ Real-world vulnerabilities **detected** and **tolerated**:
  - ➥ Memcached denial-of-service
  - ➥ Nginx stack buffer overflow
  - ➥ Apache Heartbleed

- ~~Motivation~~
- ~~Constraints of SGX enclaves~~
- ~~Design of SGXBounds~~
- ~~Implementation of SGXBounds~~
- ~~Evaluation~~

Conclusion

Conclusion

- **Security** is barrier to adoption of cloud computing

  ➥ Use **shielded execution** with Intel SGX

Conclusion

- **Security** is barrier to adoption of cloud computing
  ➥ Use **shielded execution** with Intel SGX
- Insufficient to protect against **SW vulnerabilities**
  ➥ Use **memory defense** like ASan and MPX

Conclusion

- **Security** is barrier to adoption of cloud computing
  ➥ Use **shielded execution** with Intel SGX
- Insufficient to protect against **SW vulnerabilities**
  ➥ Use **memory defense** like ASan and MPX
- ASan and MPX **perform poorly** in SGX enclaves
  ➥ Their memory assumptions are **violated in SGX**

Conclusion

- **Security** is barrier to adoption of cloud computing
  ➥ Use **shielded execution** with Intel SGX

- Insufficient to protect against **SW vulnerabilities**
  ➥ Use **memory defense** like ASan and MPX

- ASan and MPX **perform poorly** in SGX enclaves
  ➥ Their memory assumptions are **violated in SGX**

- **SGXBounds:** memory safety for shielded execution

Conclusion

- **Security** is barrier to adoption of cloud computing
  ➥ Use **shielded execution** with Intel SGX

- Insufficient to protect against **SW vulnerabilities**
  ➥ Use **memory defense** like ASan and MPX

- ASan and MPX **perform poorly** in SGX enclaves
  ➥ Their memory assumptions are **violated in SGX**

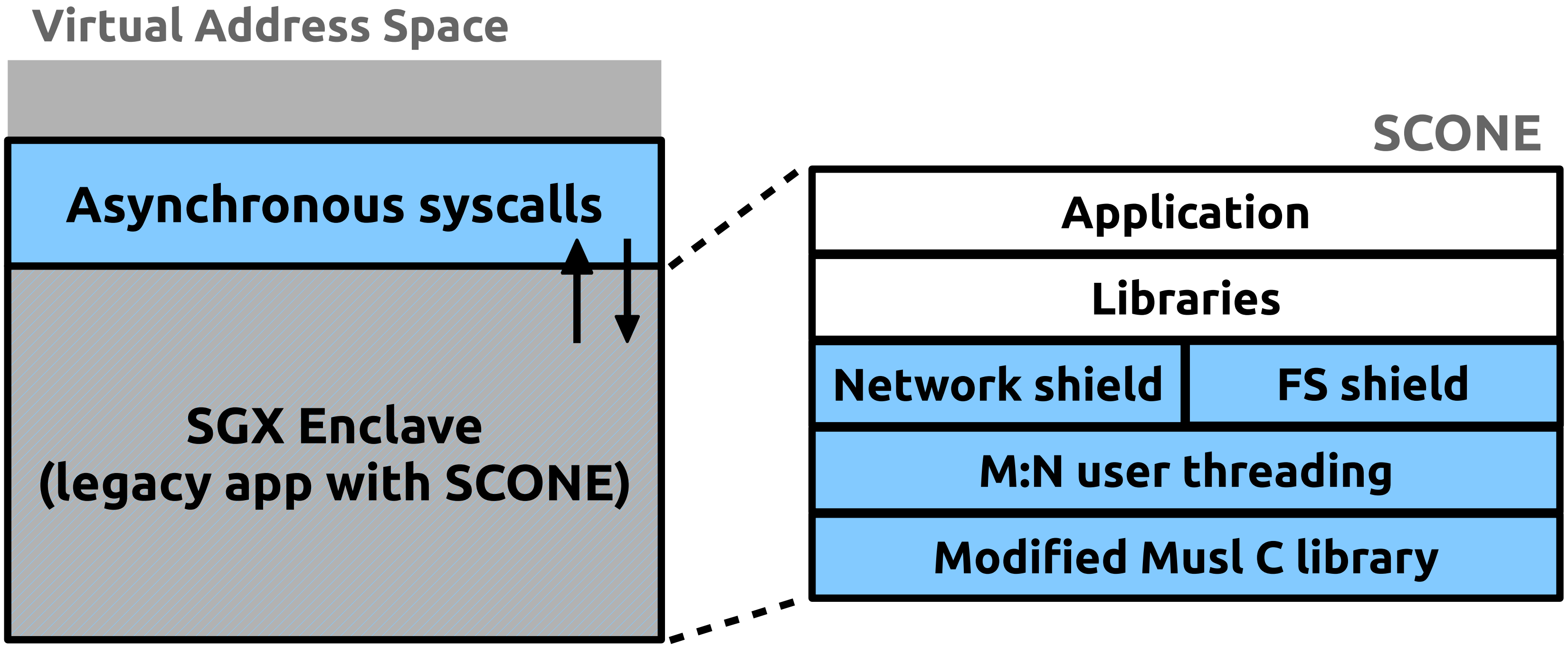
- **SGXBounds:** memory safety for shielded execution

Thank you!

dmitrii.kuvaiskii@tu-dresden.de
https://github.com/tudinfse/sgxbounds

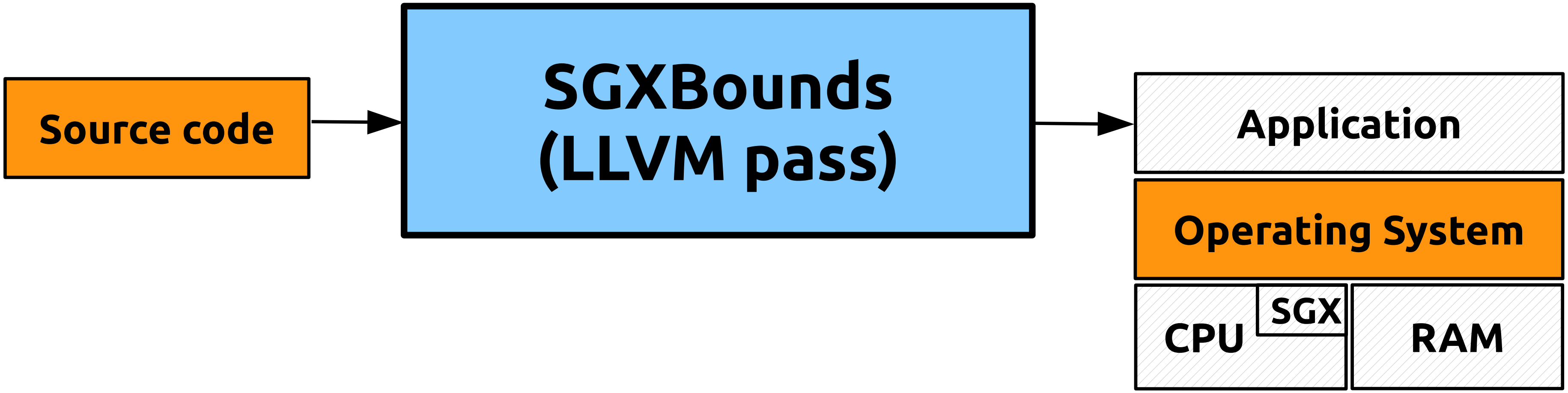
Backup slides

Intel SGX and SCONE



- V. Costan, S. Devadas. „Intel SGX Explained“. IACR Cryptology ePrint Archive '16
- S. Arnautov et al. „SCONE: Secure linux containers with Intel SGX“. OSDI'16

SGXBounds: Implementation



Native

```
a   = add x, i




store 42, a
```

SGXBounds

```
x = specify_bounds(x, x+N)
a   = add x, i
aptr = extract_ptr(a)
UB   = extract_upper_bound(a)
LB   = load_lower_bound(UB)
if (aptr < LB or aptr >= UB):
    handle_error(aptr)
store 42, a
```

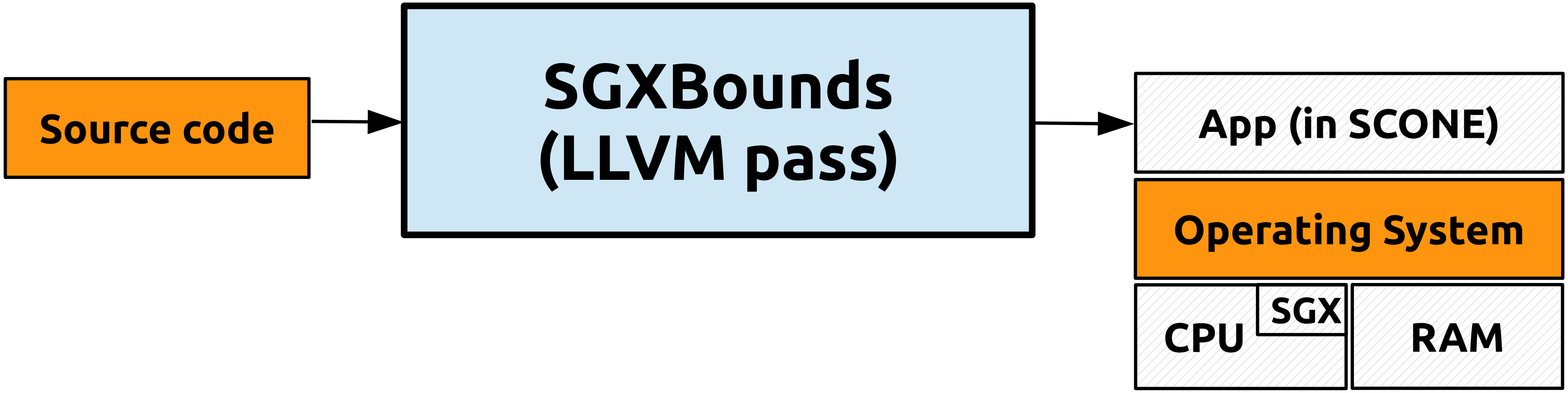
Related Work (SPEC CPU2006 outside of SGX enclave)

	Perf	Mem	Comments
Intel MPX	146%	116%	FP/FN for multithreaded
AddressSanitizer	38%	292%	–
BaggyBounds[1]	70%	12%	Not publicly available
Low-Fat Pointers[2]	54%	12%	Not publicly available
SGXBounds	55%	0%	(this work)

[1] P. Akritidis et al. „Baggy Bounds Checking: An efficient and backwards-compatible defense against out-of-bounds errors“. Usenix Security'09

[2] G. Duck et al. „Stack Bounds Protection with Low Fat Pointers“. NDSS'17

SGXBounds: Implementation



Instrumentation:

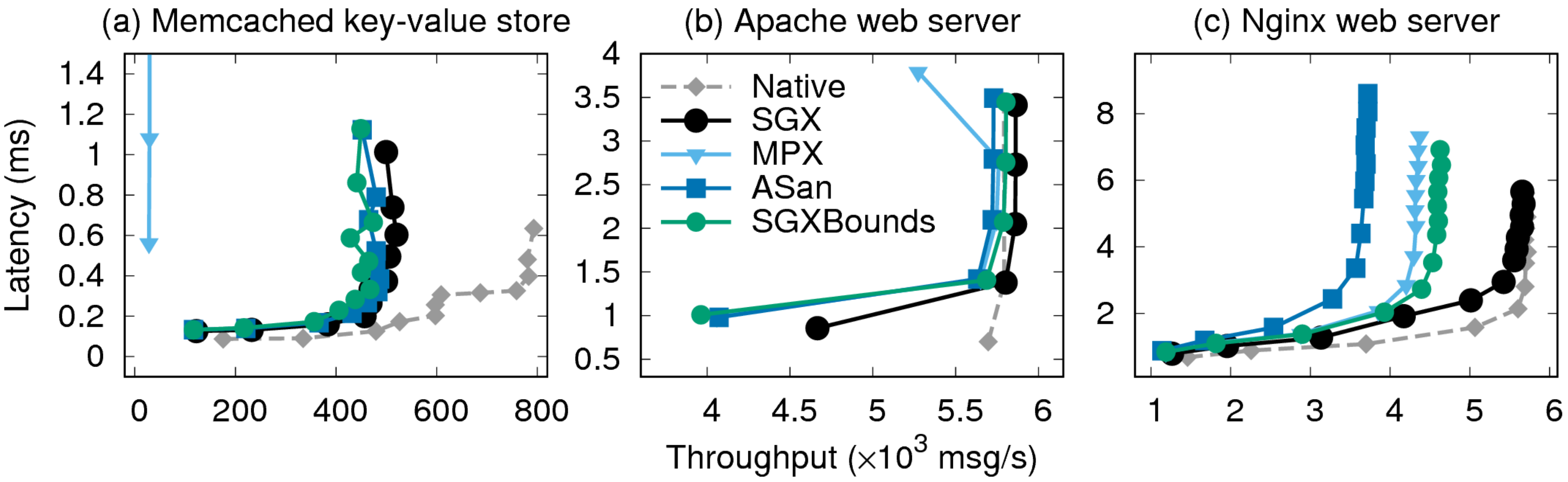
data:	**lower bound metadata** after each **allocated object**
pointers:	**upper bound metadata** in each **data pointer**
code:	**bounds-check** before each **memory access**

Security guarantees

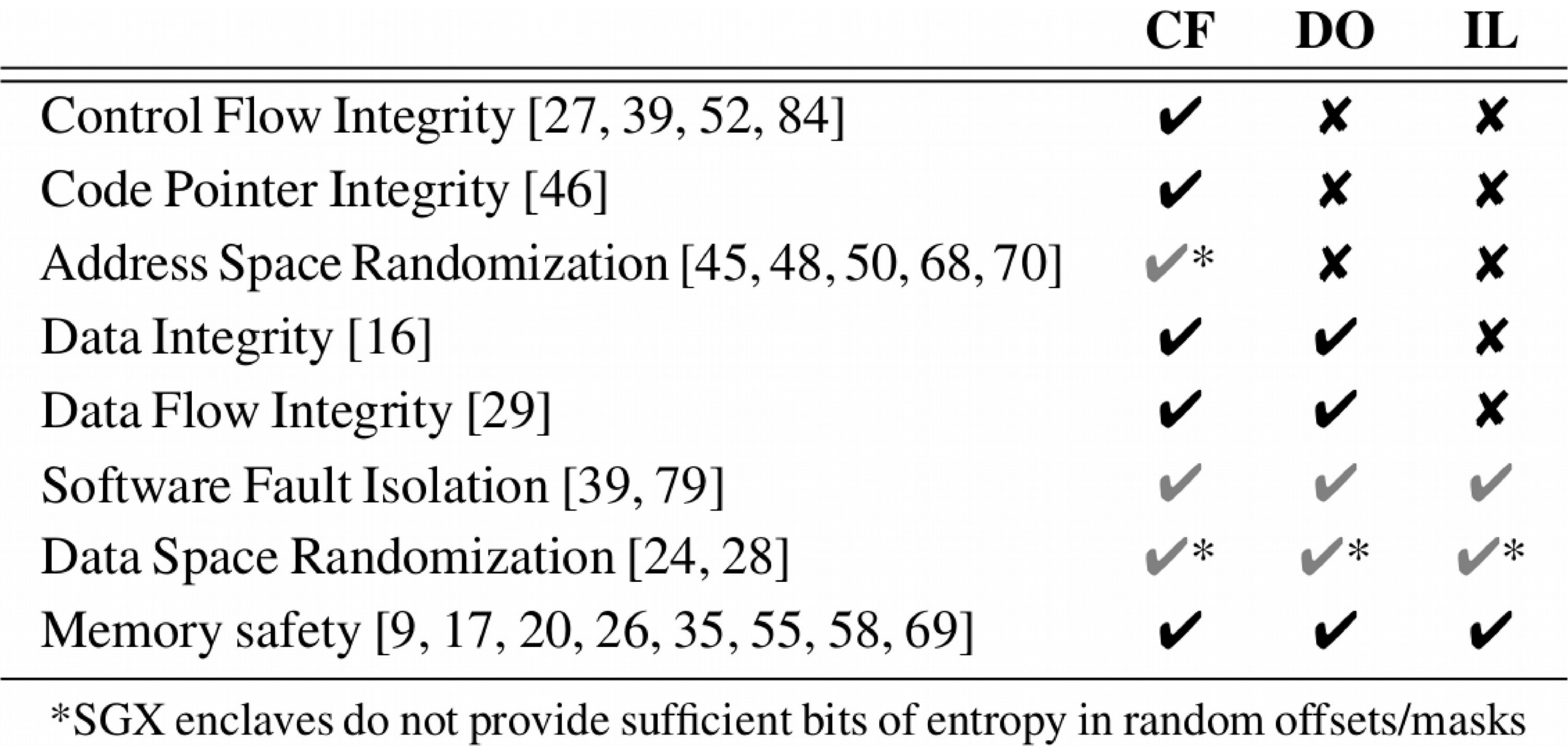
D	detected?
T	tolerated?

	MPX	ASan	SGXBounds
RIPE benchmark	2/16	8/16	8/16
Memcached CVE-2011-4971	D (T)	D (T)	D (T)
Nginx CVE-2013-2028	D (T)	D (T)	D (T)
Apache Heartbleed	D (T)	D (T)	D (T)

Case Studies: Full Picture



Classes of Defenses against Attacks

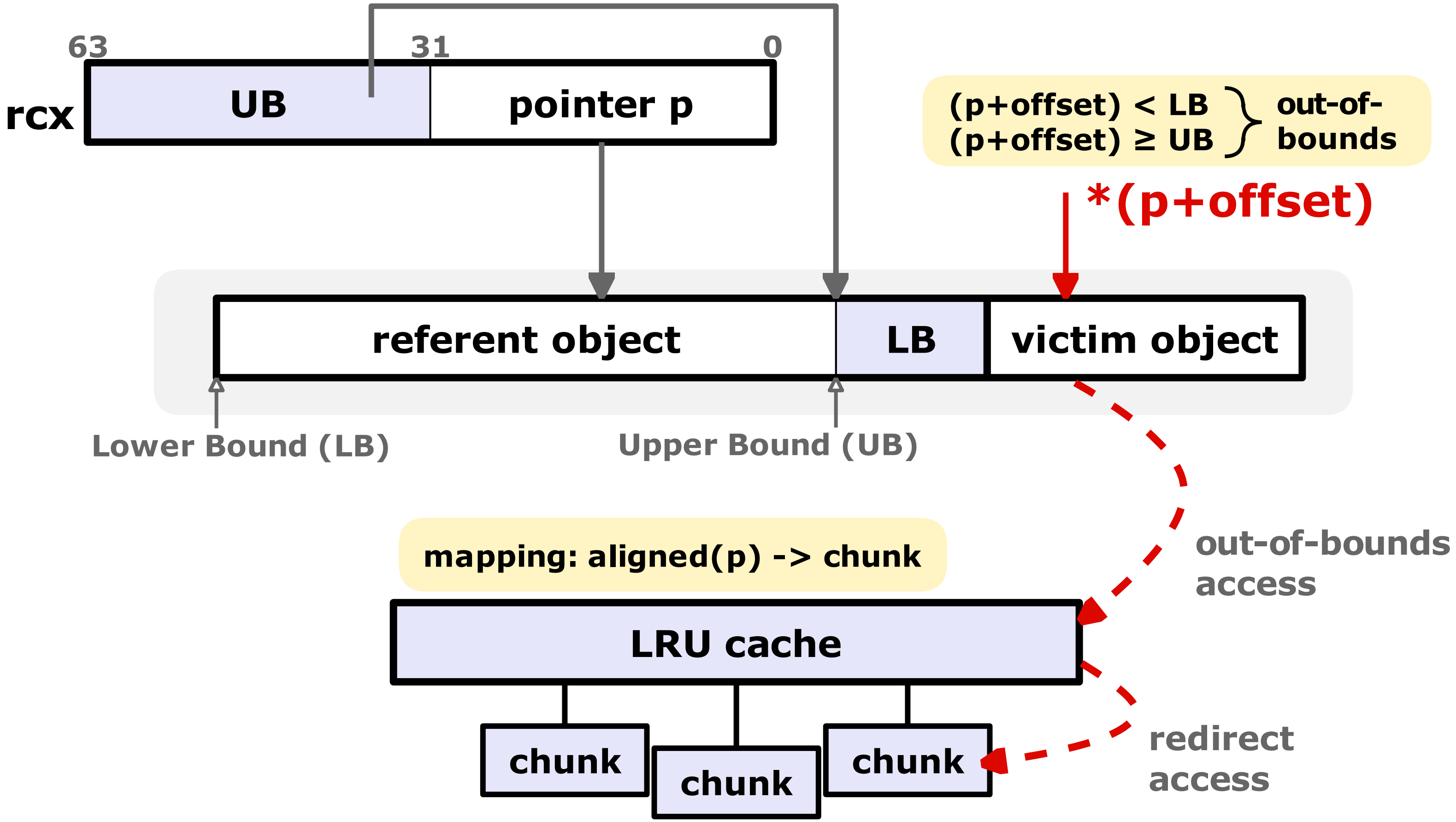


	CF	DO	IL
Control Flow Integrity [27, 39, 52, 84]	✓	✗	✗
Code Pointer Integrity [46]	✓	✗	✗
Address Space Randomization [45, 48, 50, 68, 70]	✓*	✗	✗
Data Integrity [16]	✓	✓	✗
Data Flow Integrity [29]	✓	✓	✗
Software Fault Isolation [39, 79]	✓	✓	✓
Data Space Randomization [24, 28]	✓*	✓*	✓*
Memory safety [9, 17, 20, 26, 35, 55, 58, 69]	✓	✓	✓

*SGX enclaves do not provide sufficient bits of entropy in random offsets/masks

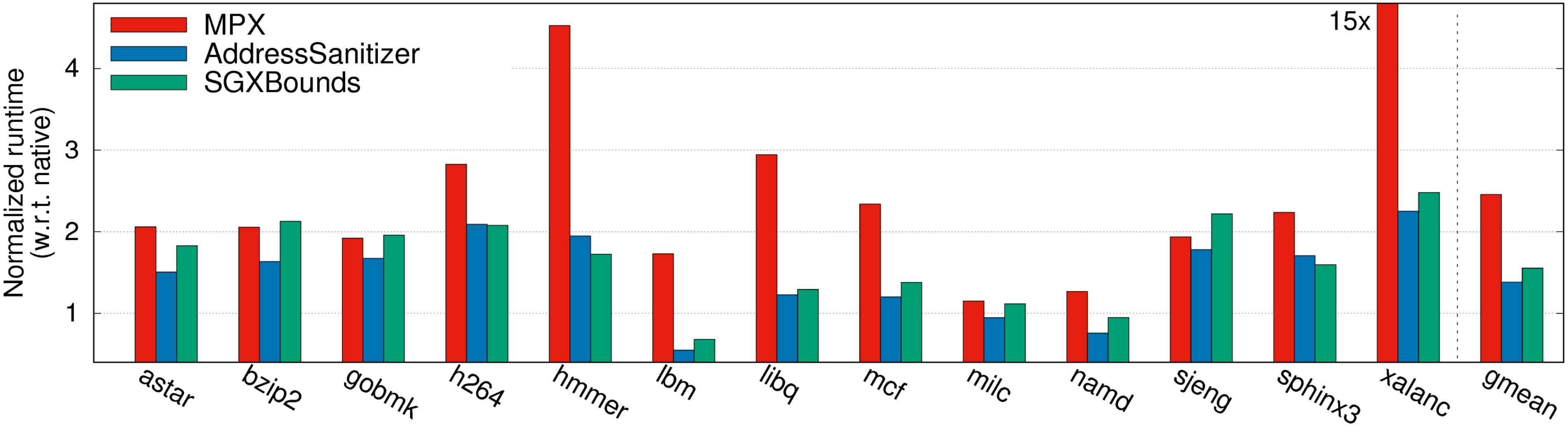
CF – control flow hijack, **DO** – data-only attack, **IL** – information leak

SGXBounds and Boundless Memory



[1] M. Rinard et al. „A dynamic technique for eliminating buffer overflow vulnerabilities (and other memory errors)“. ACSAC'04

SGXBounds: Outside of Enclaves


MPX
AddressSanitizer
SGXBounds
Normalized runtime (w.r.t. native)
15x
astar
bzip2
gobmk
h264
hmmer
lbm
libq
mcf
milc
namd
sjeng
sphinx3
xalanc
gmean