Graphene-SGX

A Practical Library OS for Unmodified Applications on SGX

Chia-Che Tsai Donald E. Porter Mona Vij

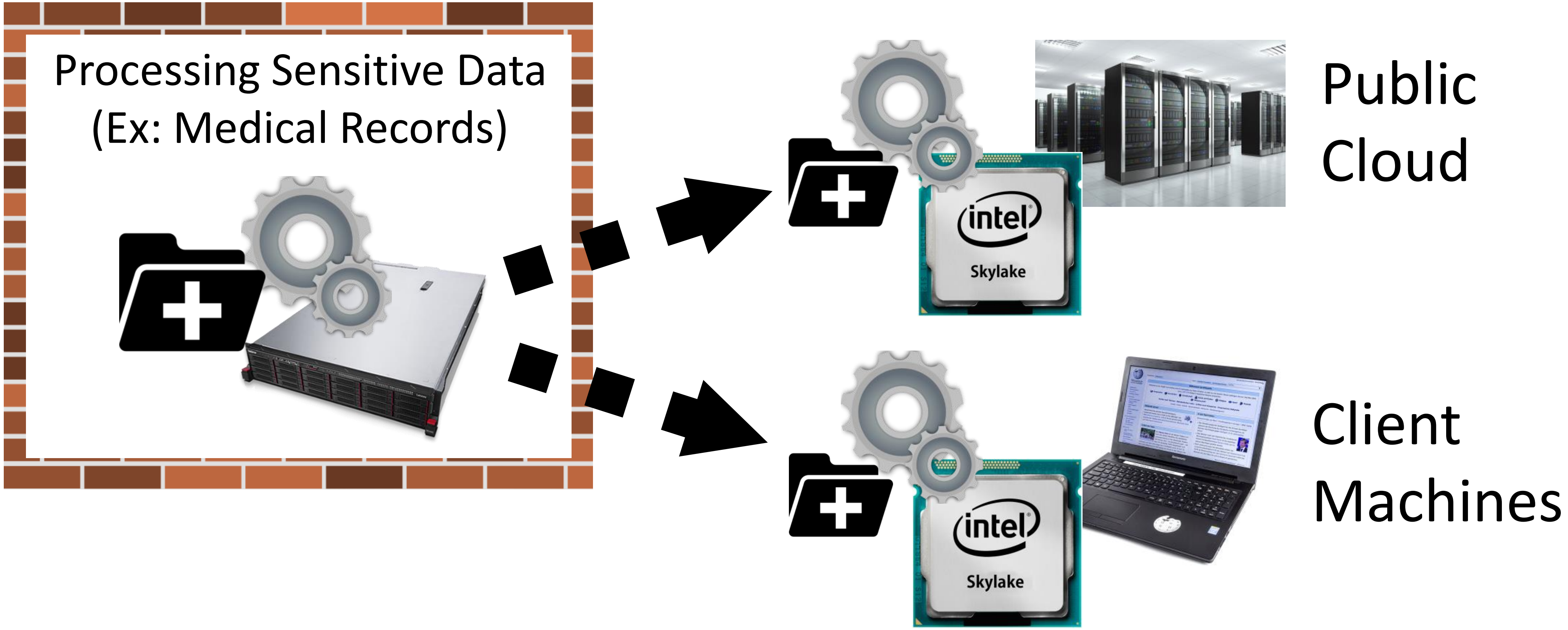
Stony Brook University

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Fortanix

intel

Intel SGX: Trusted Execution on Untrusted Hosts



Processing Sensitive Data
(Ex: Medical Records)

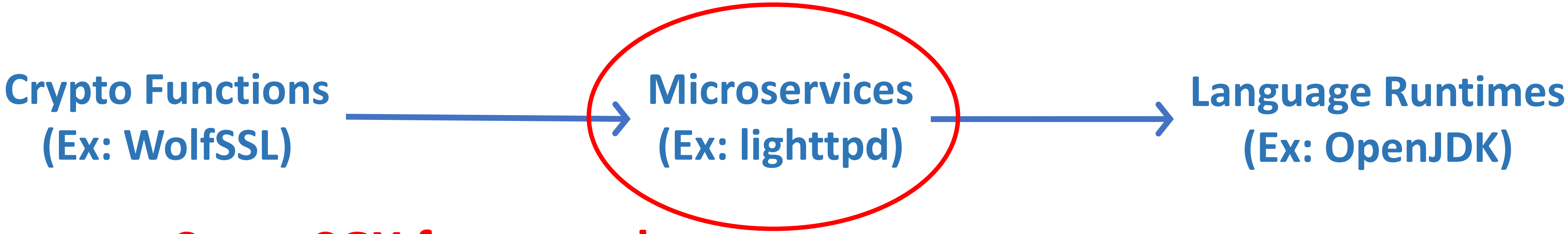
Public
Cloud

Client
Machines

App confidentiality & integrity on machines you have no control

Porting Apps to SGX is Not Exactly Painless

- OS functionality available but not trusted
- Porting: novice → hell



Crypto Functions (Ex: WolfSSL)

Microservices (Ex: lighttpd)

Language Runtimes (Ex: OpenJDK)

Some SGX frameworks (SCONE/Panoply) target here

Still “some” porting effort (Ex: recompiling)

An effortless option for wide-ranged Ubuntu apps?

Open SGX framework for Unmodified Linux Apps

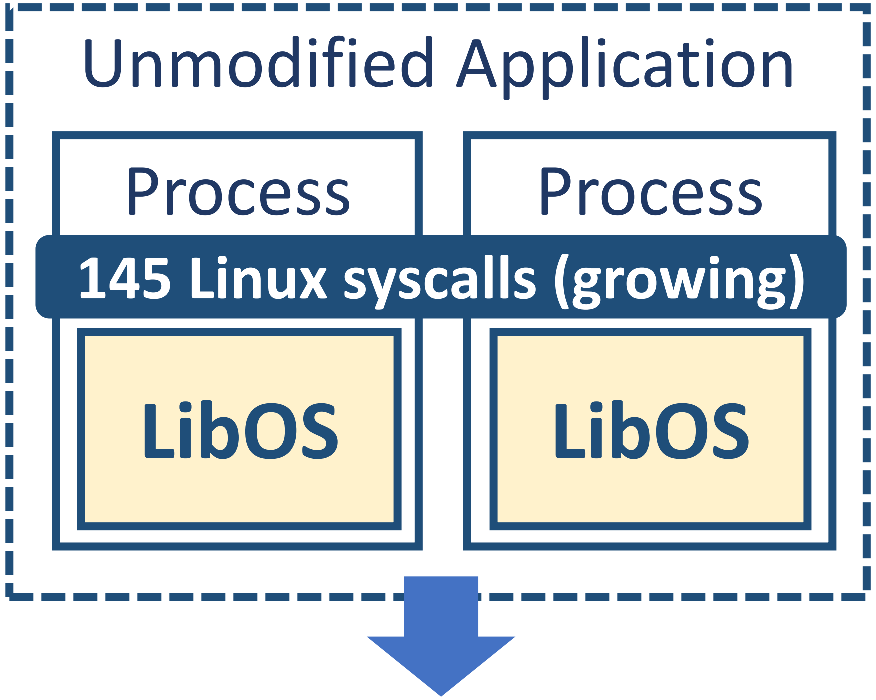
- **Graphene-SGX**:
  - No reprogramming or recompiling
  - Servers / Command-line apps / Runtimes
    (Apache, NGINX, GCC, R, Python, OpenJDK, Memcached, …)
  - Multi-process APIs (fork, IPC, …)
  - **Not meant to be perfect, but a quick, practical option**
    (or to avoid app changes)

Talk Outline

- How does Graphene-SGX protect unmodified applications?
- Why should you try Graphene-SGX?
- What is the right way for porting applications to SGX?

The Graphene LibOS Project [Eurosys14]

- An open libOS for reusing Linux applications (**github.com/oscarlab/graphene**)
  - Inspired by Drawbridge[ASPLOS11] and Haven[OSDI14]
  - Gradually adopted by labs / industry
  - Active development & tech support (doing our best!)



Unmodified Application

Process

Process

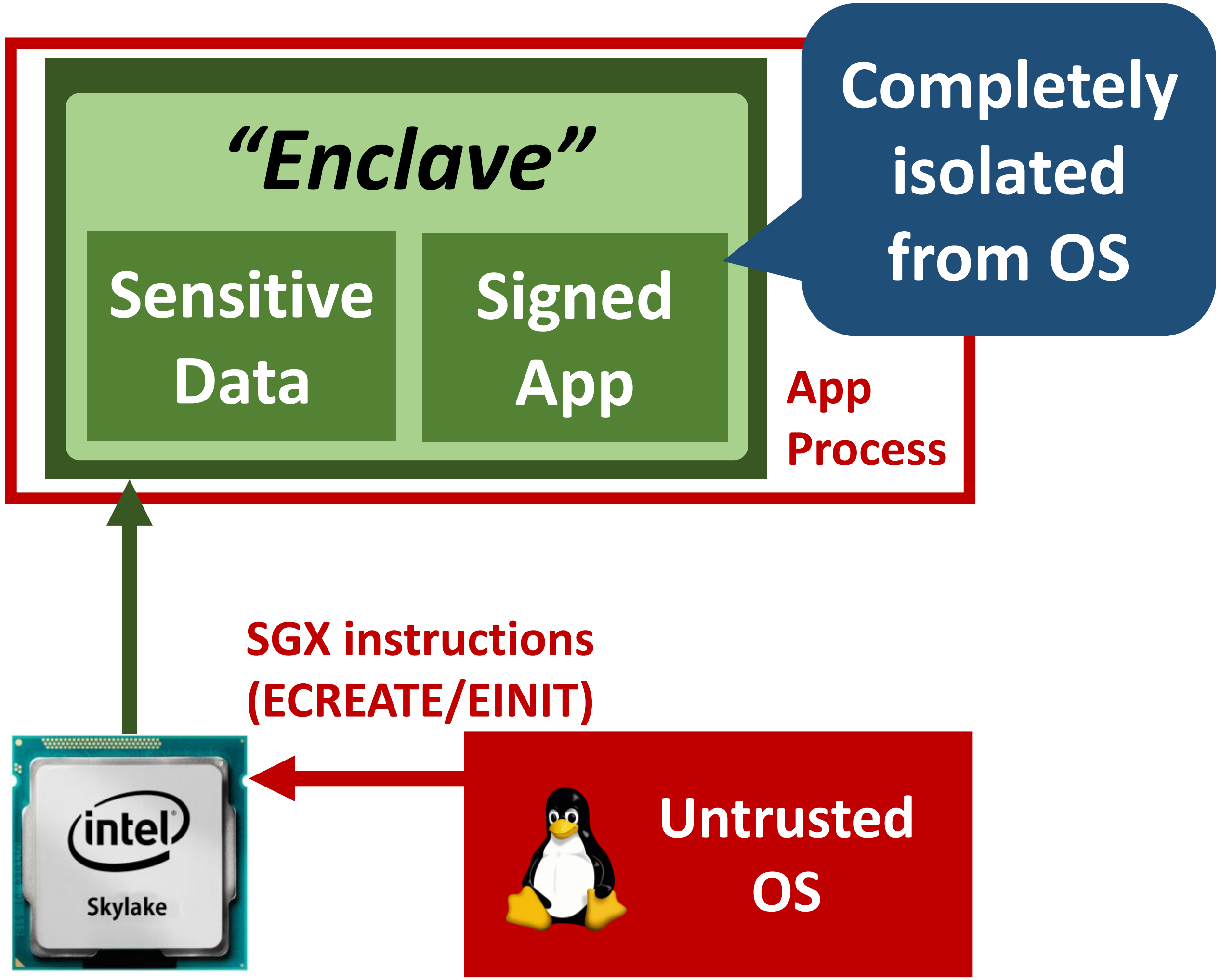
145 Linux syscalls (growing)

LibOS

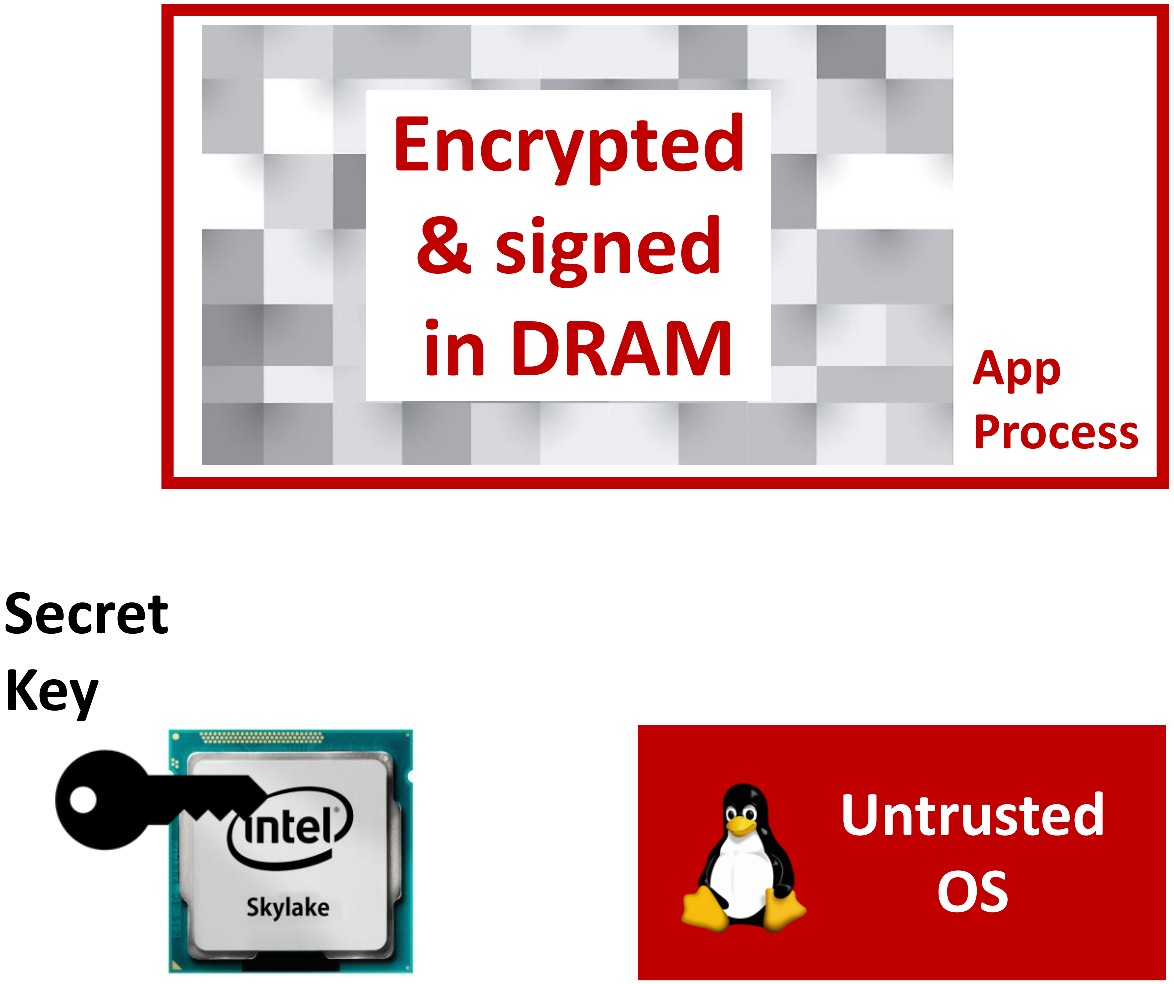
LibOS

Easy to port to new OS/platform

Intel SGX (Software Guard Extensions)



Intel SGX (Software Guard Extensions)



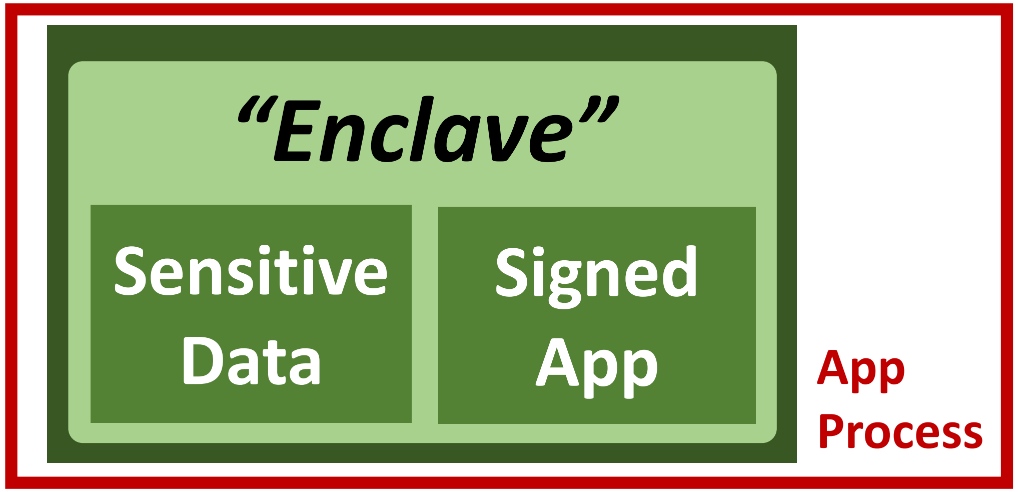
Encrypted & signed in DRAM

App Process

Secret Key

Untrusted OS

Intel SGX (Software Guard Extensions)



Enclave app requirements:

1. Signed initial code
2. No direct syscalls
3. Checking untrusted inputs

Unmodified Linux app:

(1) Dynamic linked

(2) Built with syscall usage

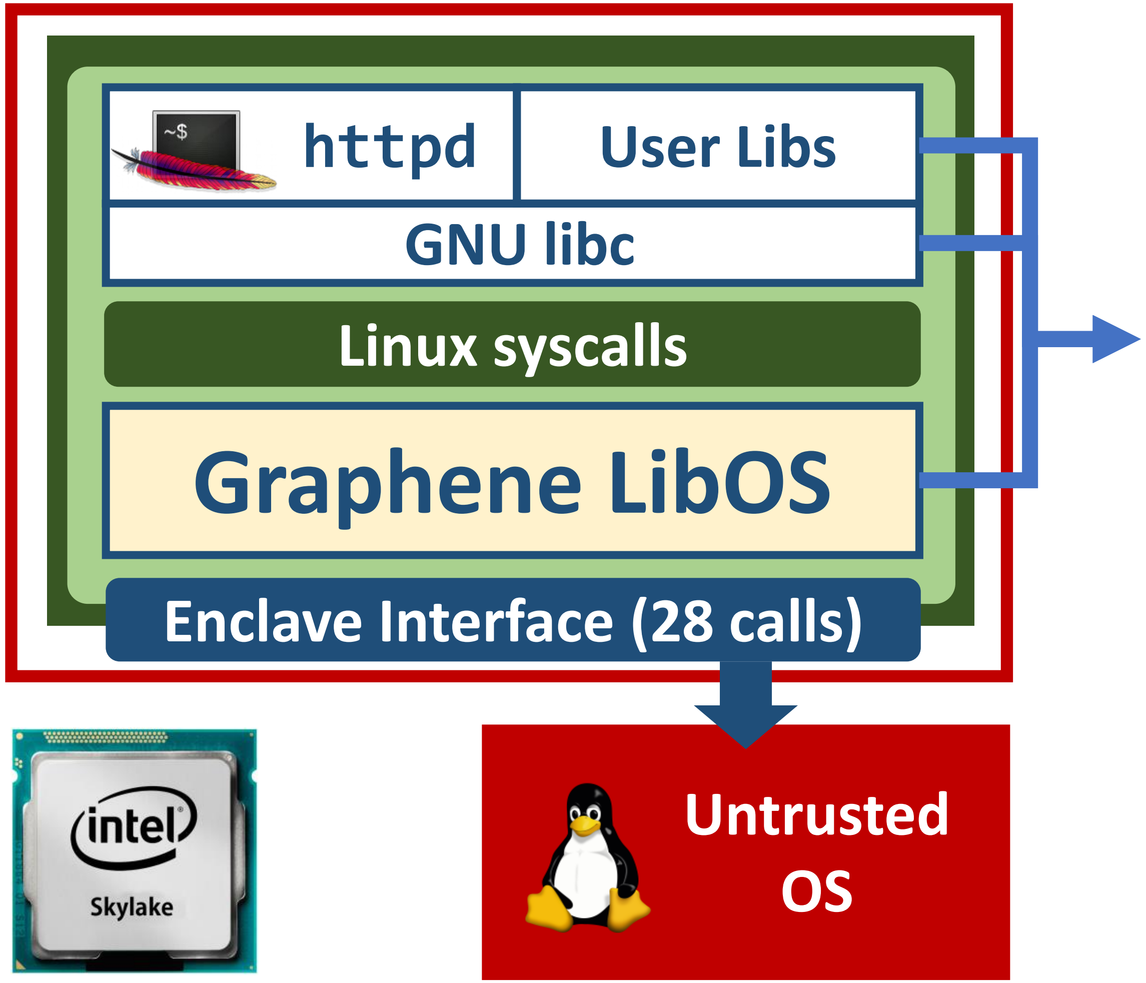
Running Unmodified App with Graphene-SGX

```
$ SGX=1 ./pal_loader httpd [args]
```

Graphene Loader

Untrusted OS

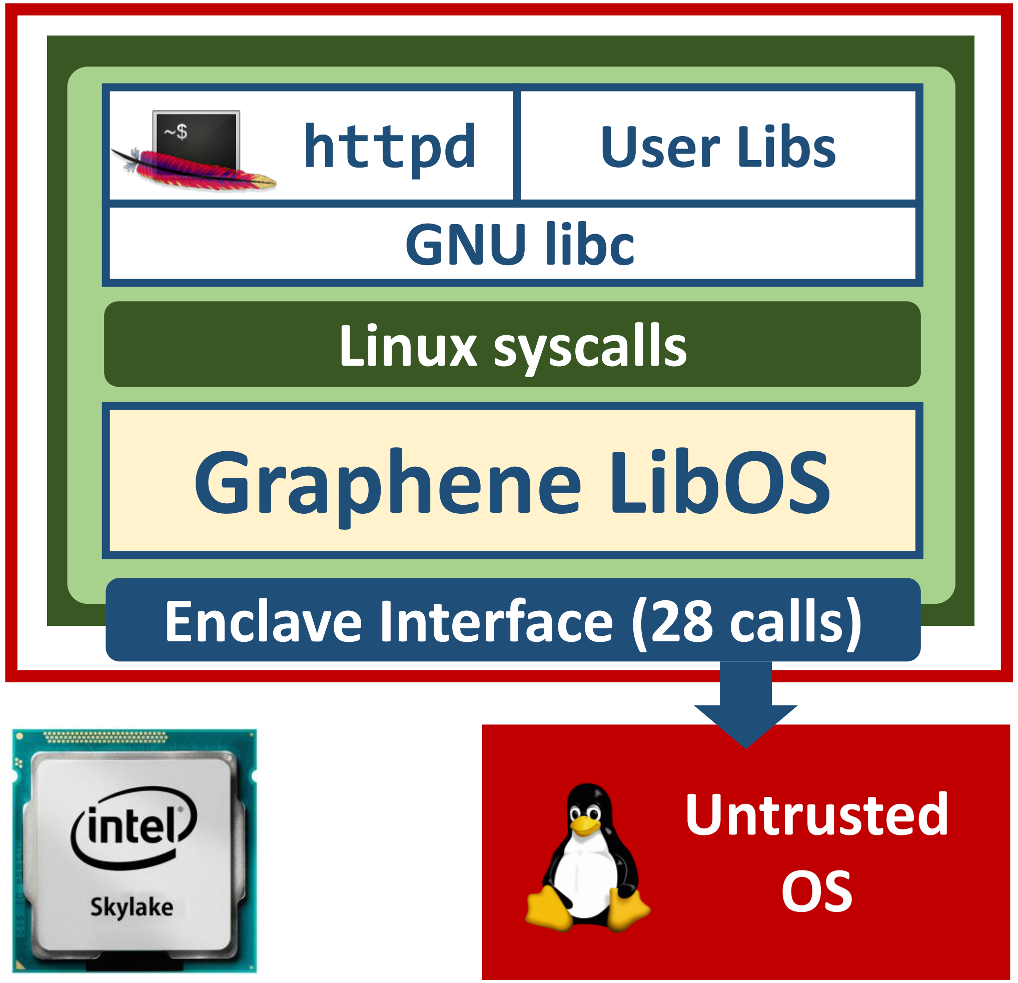
Running Unmodified App with Graphene-SGX



Signed by developers as a CPU-verifiable signature

(Signing tool provided)

Running Unmodified App with Graphene-SGX



Enclave app requirements:

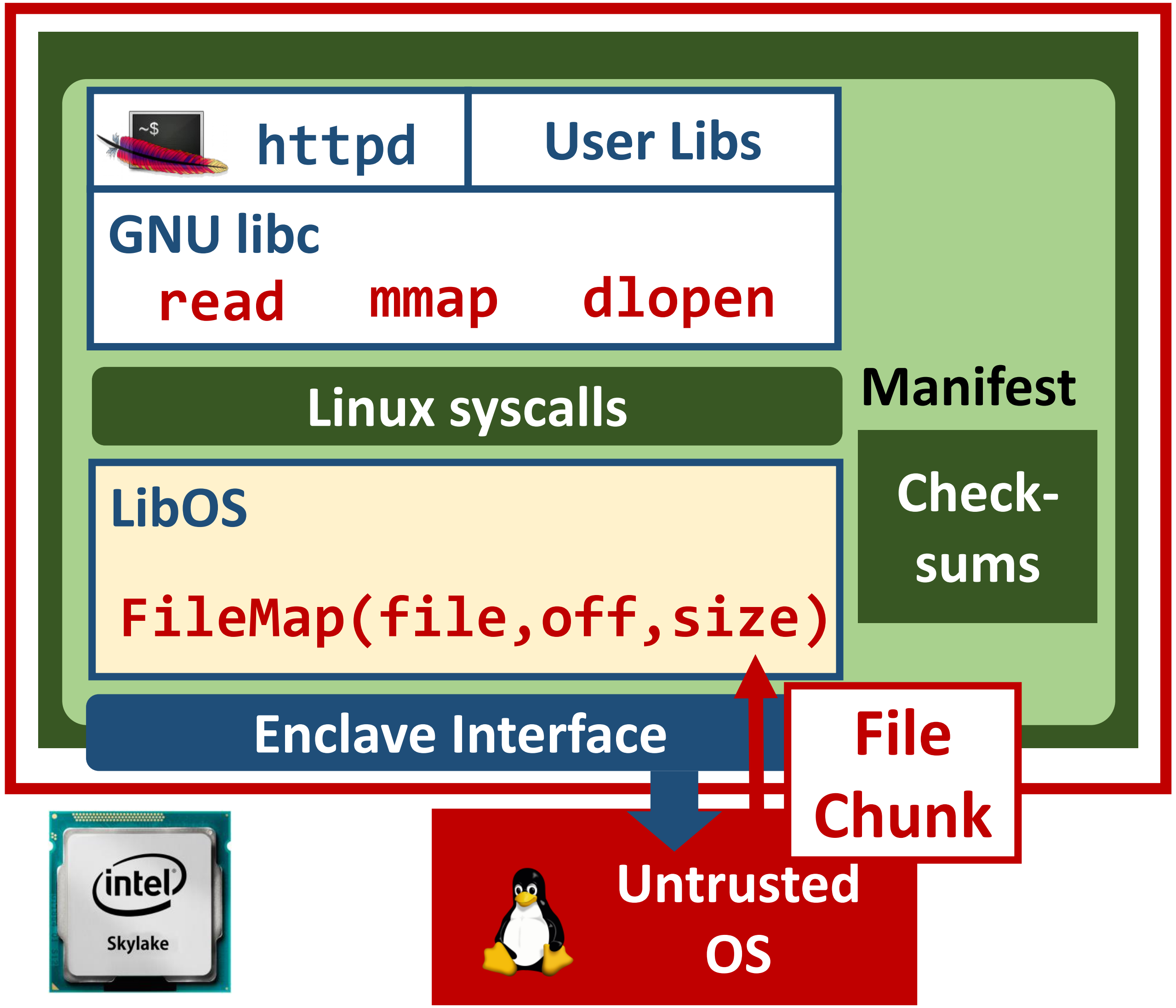
1. Signed initial code ✓
2. No direct syscalls ✓
3. Checking untrusted inputs

key research problem

Checking Untrusted Inputs from the OS

- Checking untrusted syscalls is subtle [Checkoway, 2013]
- Graphene-SGX:
  - Narrowing to a fixed interface (28 calls)
  - **Redefining an interface suitable for checking**
- Examples:
  - Reading an integrity-sensitive file (Ex: library/script/config)
  - See paper: multi-process APIs

Ex: Reading an Integrity-Sensitive File



- Ask for explicit inputs
- Checksums given in a signed “manifest”
- Copy & verify in enclave

Checking All 28 Enclave Calls

Examples	#	Result	Explanation
(1) Reading a file (2) Inter-proc coordination	18	Fully Checked	(1) File checksums (2) CPU attest. + crypto: inter-proc TLS connection
Yielding a thread	6	Benign	Nothing to check
(1) Polling handles (2) File attributes	4	Unchecked	Future work

Summary

- Graphene-SGX turns an unmodified app into enclave app
  - A app-specific signature authenticating all binaries
  - Syscalls implemented inside enclaves
  - Narrowing & redefining untrusted OS inputs to checkable values

Why (and When) You Should Try Graphene-SGX

- Unmodified apps / needs dynamic loading
- When alternatives don’t offer OS functionality you want
- Graphene-SGX:
  - Rich OS functionality (145 syscalls so far)
  - Blow up enclave size & TCB (trusted computing base)?
  - Performance?

Comparison with Other SGX Frameworks

	Graphene-SGX	SCONE [OSDI16]	Panoply [NDSS17]
Approach	**LibOS**	**“Shim” Layers: redirect & check system APIs**	
Functionality vs checks	Can grow without extending checks	Using more system APIs = more checks	

Trusted Computing Base

	Graphene-SGX	SCONE [OSDI16]	Panoply [NDSS17]
LibOS/shim	**53 kLoC**	**97 kLoC**	**10kLoC**
Choice of libc	GNU libC (1.1 MLoC)	musl (88 kLoC)	No libc in enclave

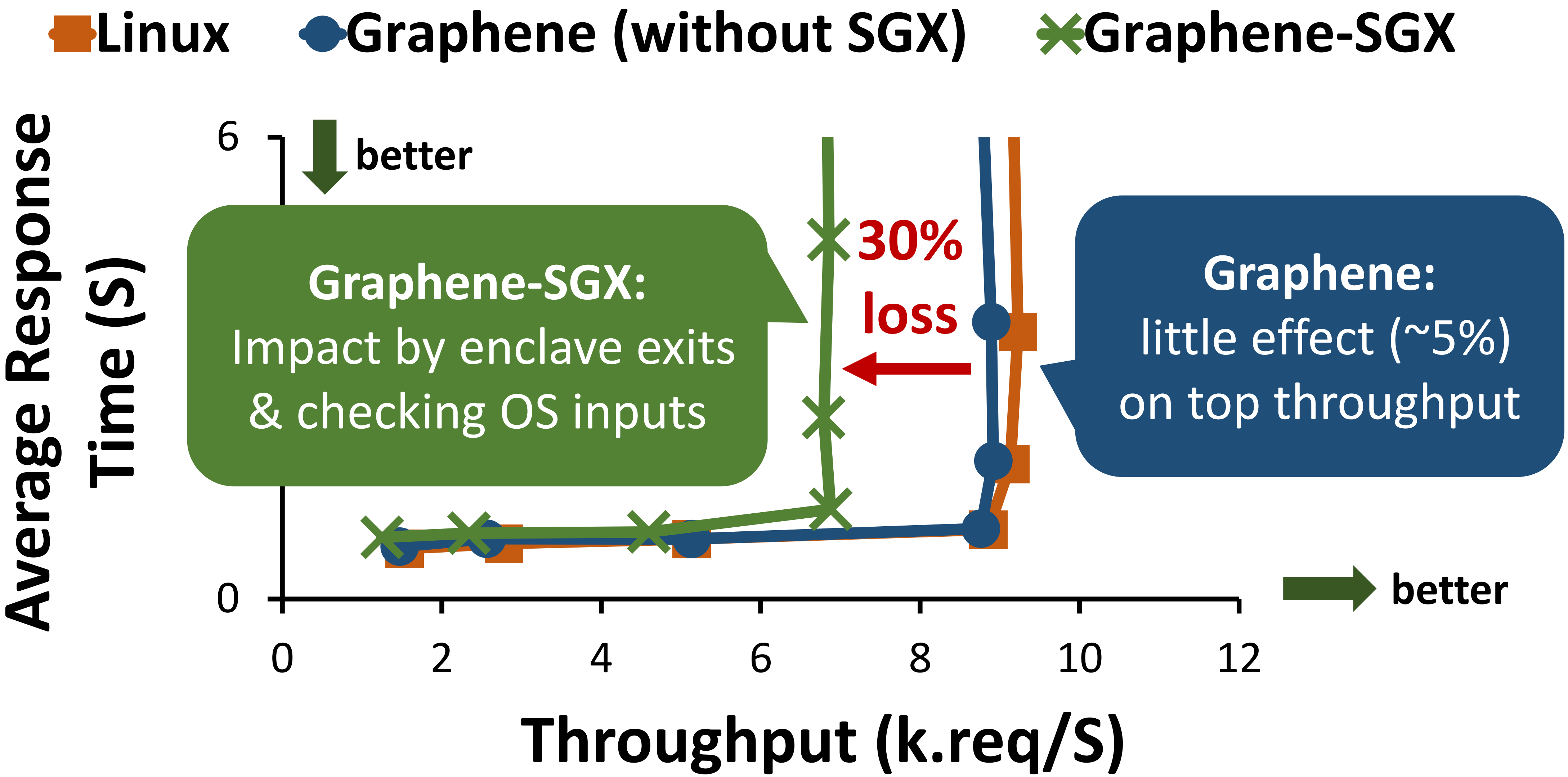
Not fundamental to libOS, but more by the choice of libc

Graphene-SGX Performance

- Baselines: Linux, Graphene (without SGX)
- Workloads:
  - Server: **Apache with 5 worker processes**
  - Command-line: **R benchmarks**
- Evaluation Setup:

4-core 3.20 GHz Intel i5 CPU + 8 GB RAM

Apache with 5 Processes (w/ IPC Semaphore)

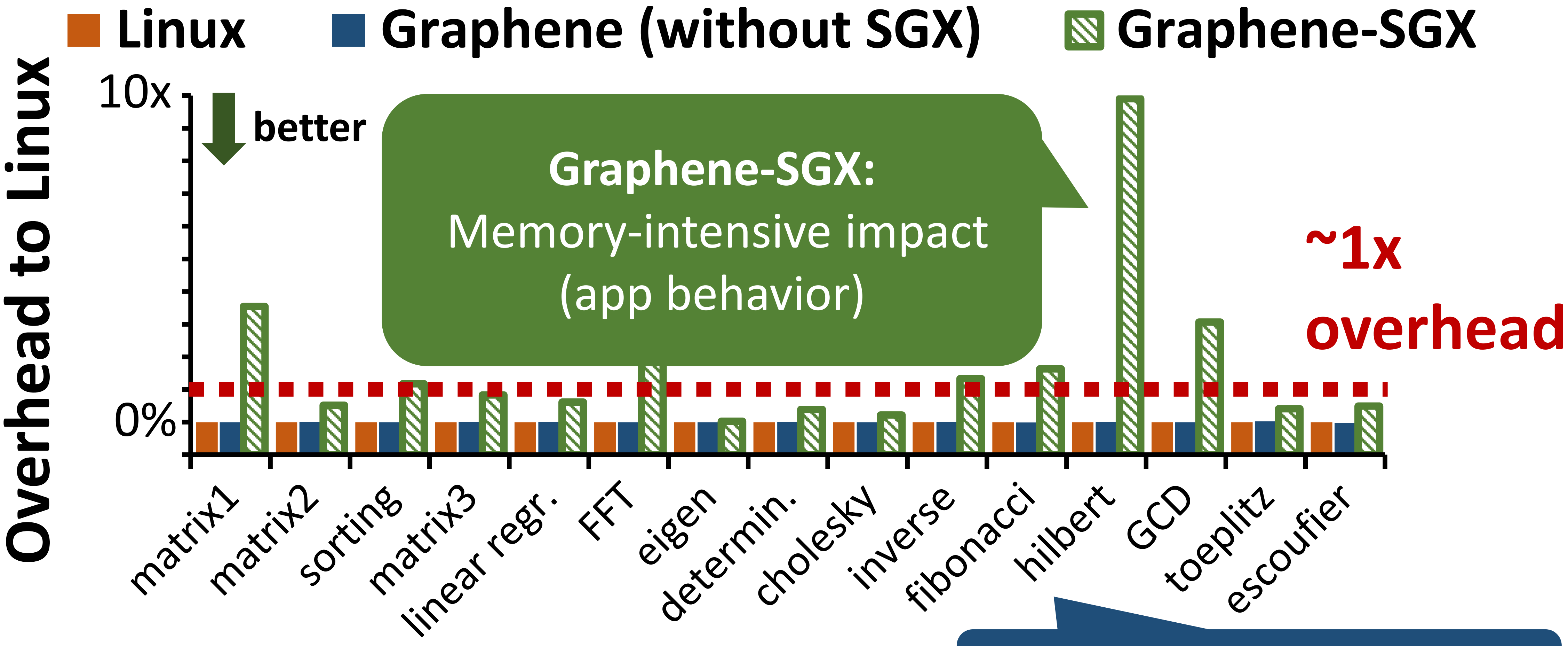


Graphene-SGX:
Impact by enclave exits
& checking OS inputs

30% loss

Graphene:
little effect (~5%)
on top throughput

R Benchmarks



Graphene-SGX:
Memory-intensive impact
(app behavior)

Graphene: ~0% overhead

Graphene-SGX Performance Discussion

- Latency overhead less than ~1x unless memory-intensive
- LibOS memory cost only 5-15 MB
- Cause:
  - Enclave exits & checks (can improve)
  - App memory usage (reduce with configuration / partitioning)

In the End: A Developer’s Guide for SGX Porting

1. Explore / POC with Graphene-SGX

2.
    - Compile out code & syscalls
    - SCONE / Panoply
    - Other tools: Eleos, T-SGX

3.
    - Partitioning (Glamdring)
    - Optimize performance & security

- Keep safe interface to OS
- Reduce memory footprint & enclave exits
- Take care of vulnerabilities (side channels!)

Conclusion

Graphene-SGX — quick, practical Linux-to-SGX porting option

- **Usability:** Rich Linux functionality with multi-process
- **Performance:** Less than ~1x overheads (normal cases)
- **Security:** (1) Reduce OS interaction to checkable services
  (2) LibOS TCB comparable to other options

Graphene library OS: github.com/oscarlab/graphene

(chitsai@cs.stonybrook.edu)

Stony Brook University

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Fortanix

intel