Panoply: Low-TCB Linux Applications With SGX Enclaves

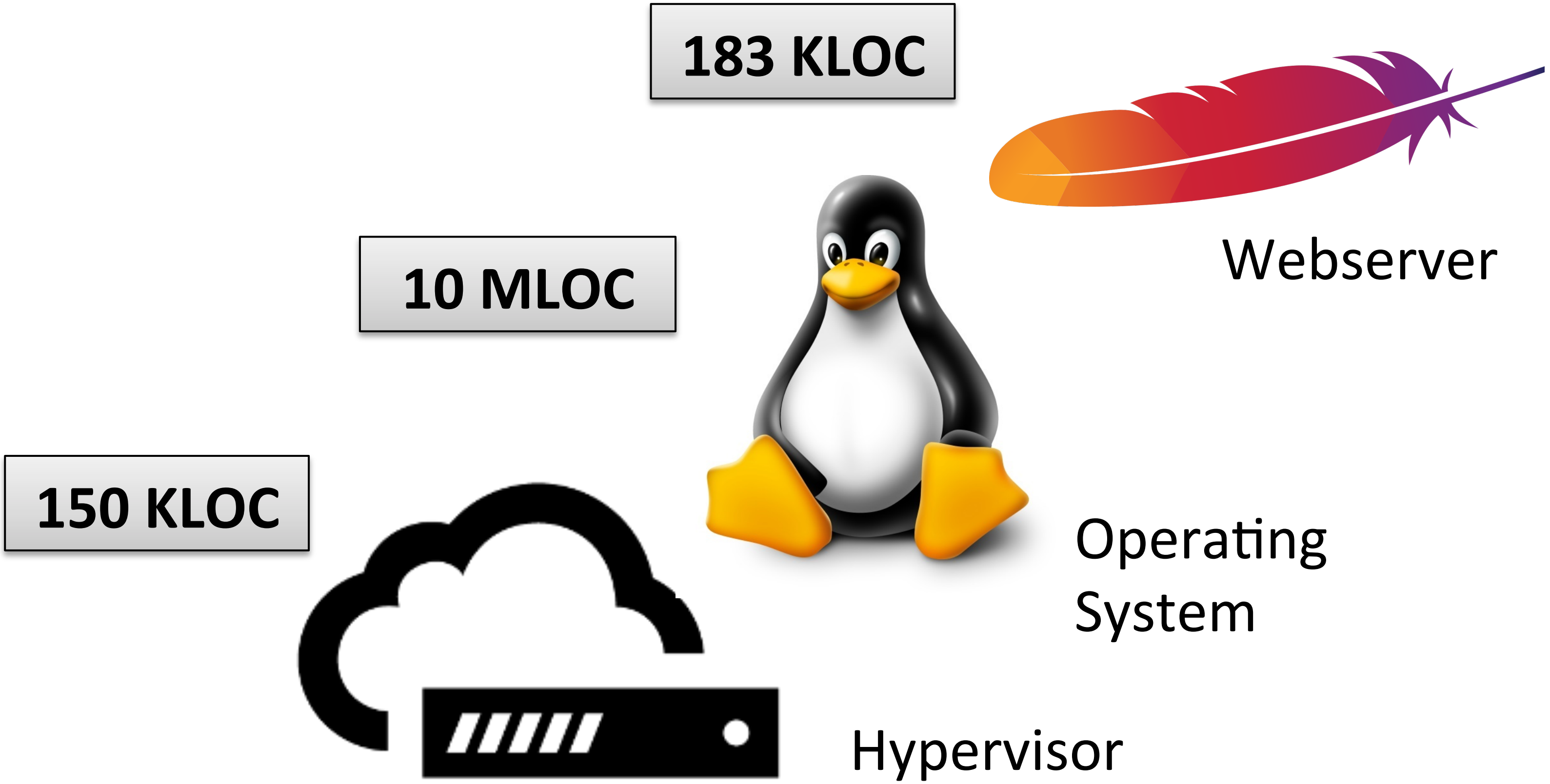
Shweta Shinde Dat Le Tien

Shruti Tople Prateek Saxena

National University of Singapore

TCB: Hosting a Web Server

Current systems have a large TCB



183 KLOC

Webserver

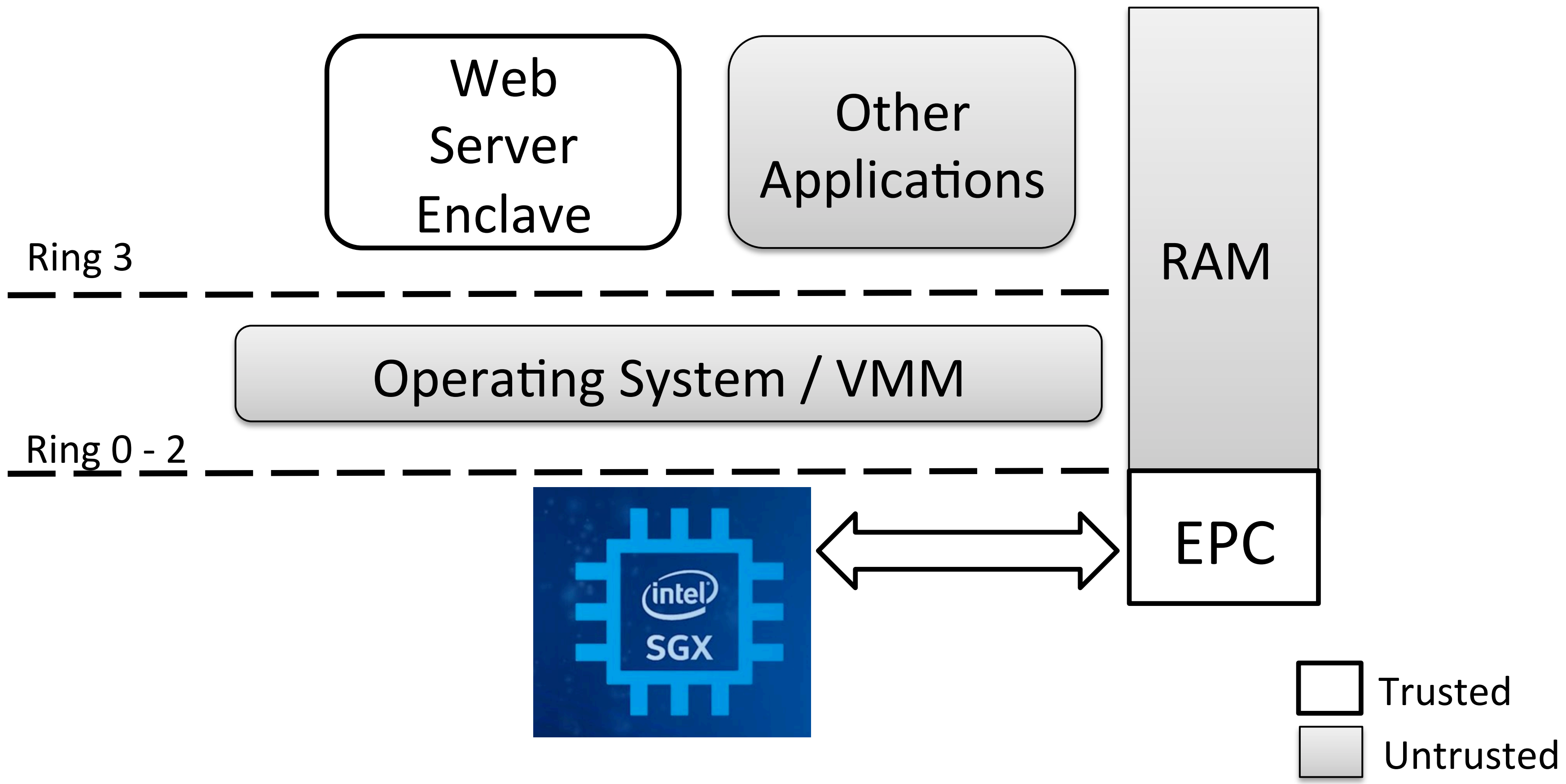
10 MLOC

Operating System

150 KLOC

Hypervisor

SGX: Hardware-root of Trust



Web Server Enclave

Other Applications

RAM

Ring 3

Operating System / VMM

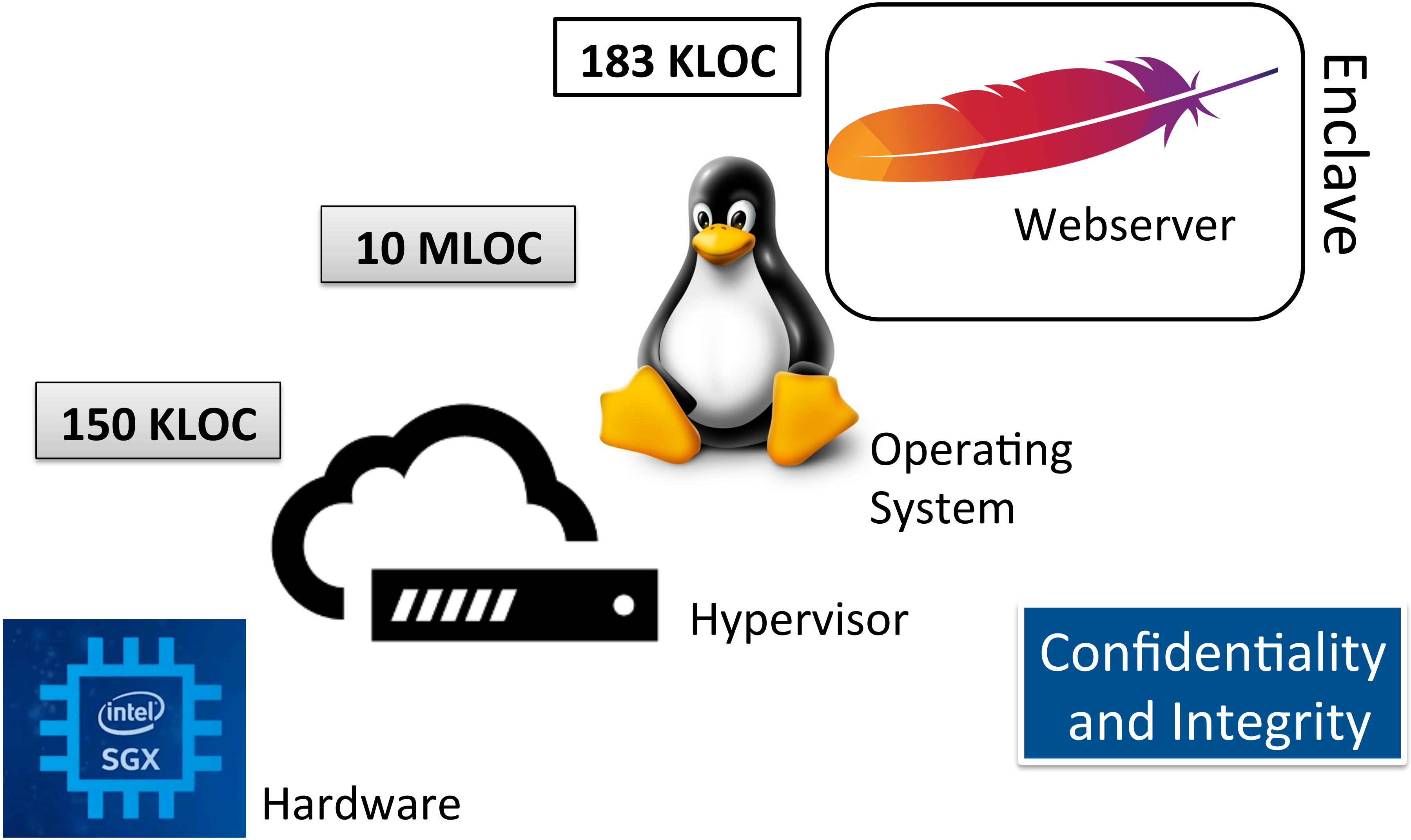
Ring 0 - 2

EPC

□ Trusted

▨ Untrusted

SGX: Hardware-root of Trust



183 KLOC

Webserver

Enclave

10 MLOC

Operating System

150 KLOC

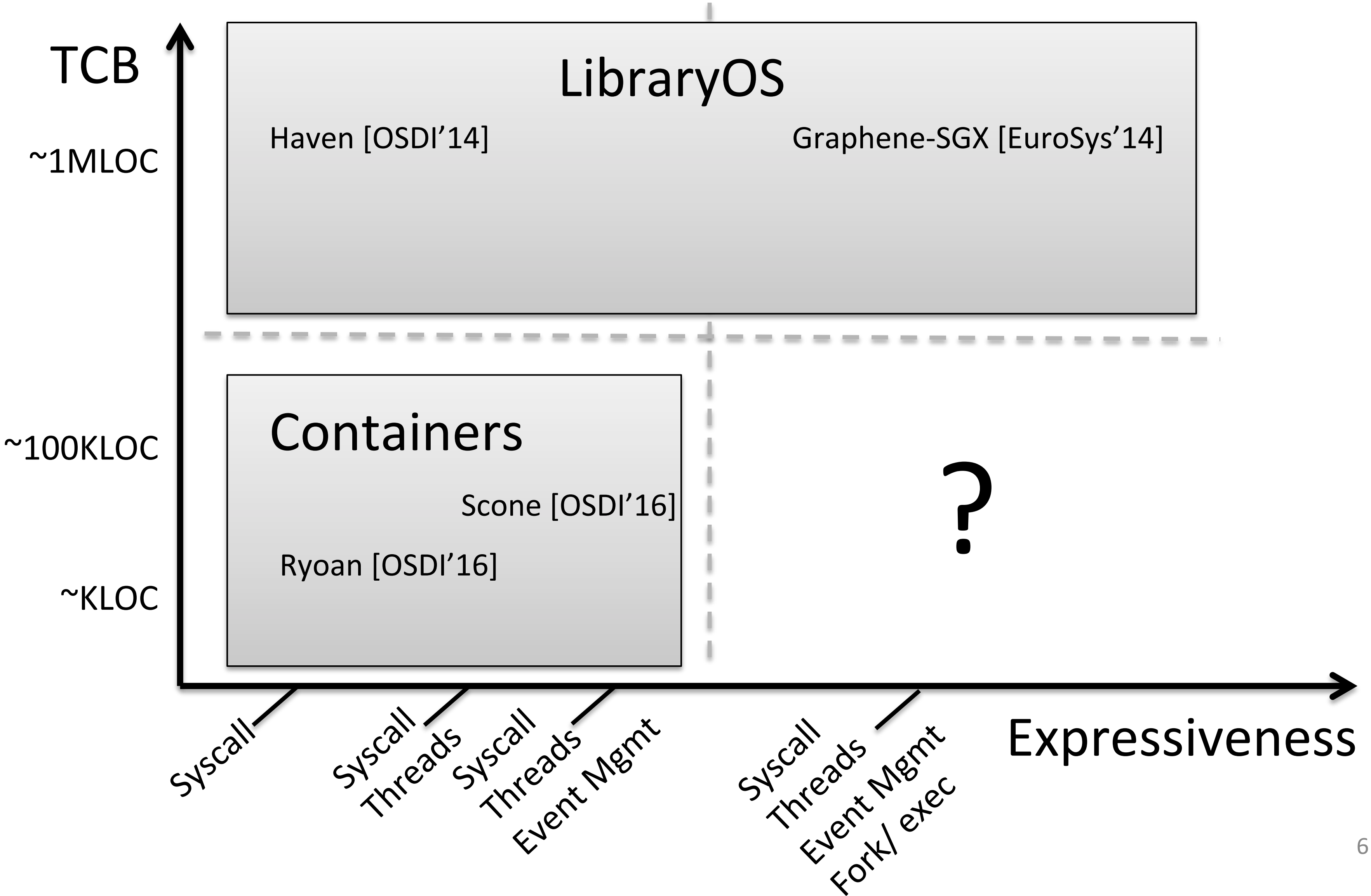
Hypervisor

Hardware

Confidentiality and Integrity

..but limits the expressiveness of the applications (e.g., no syscalls)

TCB & Expressiveness Trade-off

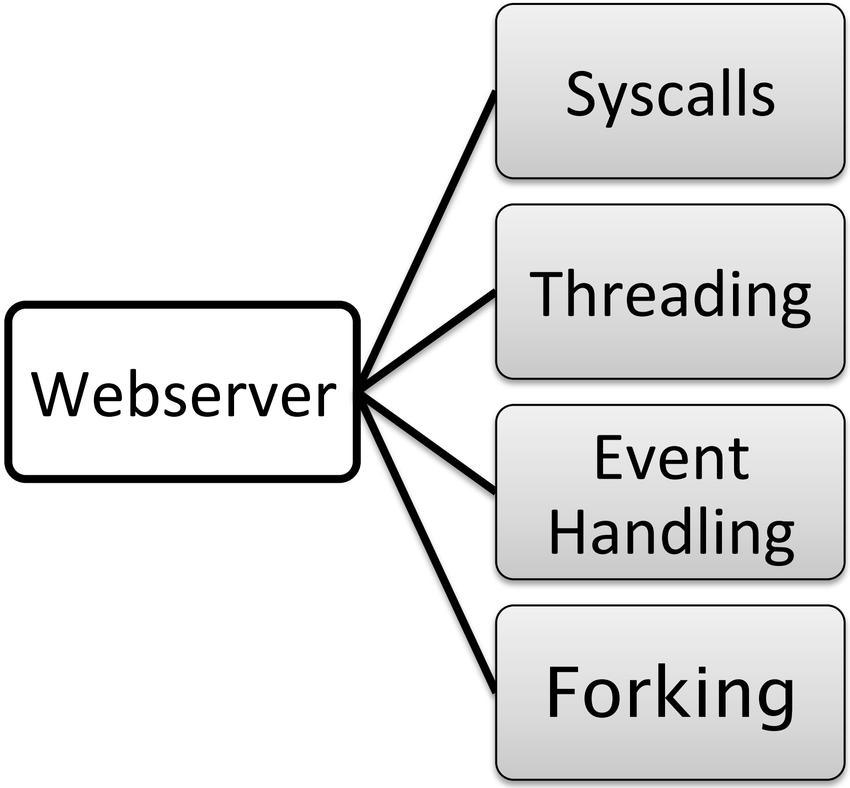


Contributions

- **Panoply**
  - Expressiveness: **All standard POSIX APIs**
  - Low TCB: **2 orders** of magnitude smaller than LibraryOS
  - Library-enclaves for fine-grained TCB
- **Evaluation**
  - Absolute **24%** and **5-10%** compared to LibraryOS

Problem

Challenge I: Expressiveness vs TCB



Webserver

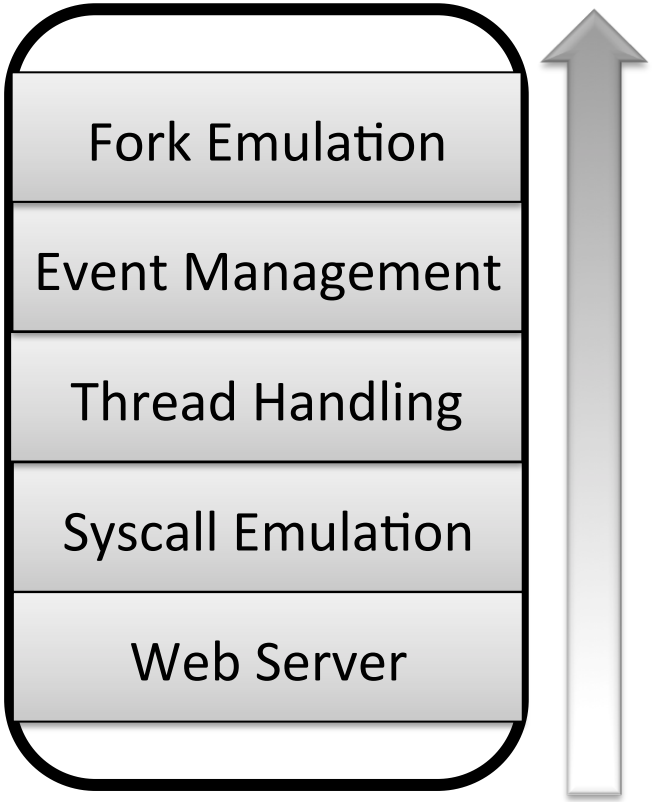
Syscalls

Threading

Event Handling

Forking

Legacy Application Design



Fork Emulation

Event Management

Thread Handling

Syscall Emulation

Web Server

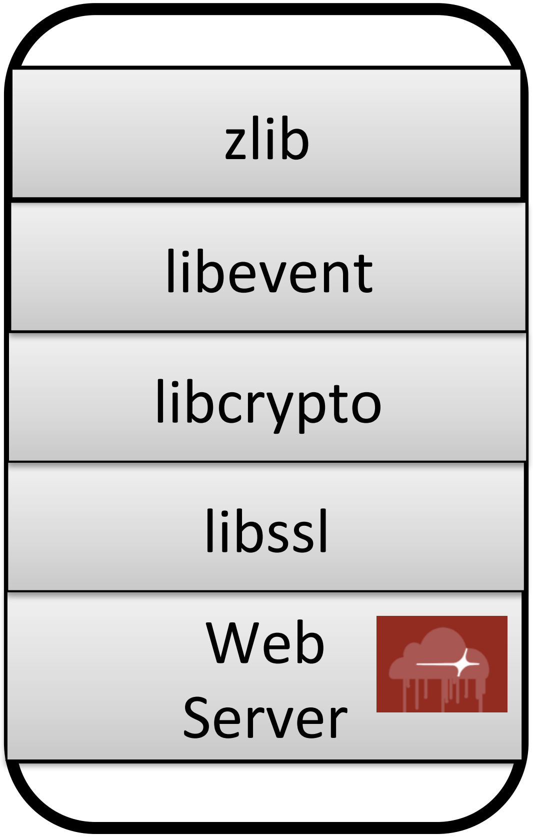
Enclave

TCB

Challenge I: Expressiveness vs. TCB

Expressiveness $\propto$ TCB

Challenge II: Multi-Enclave Applications



zlib

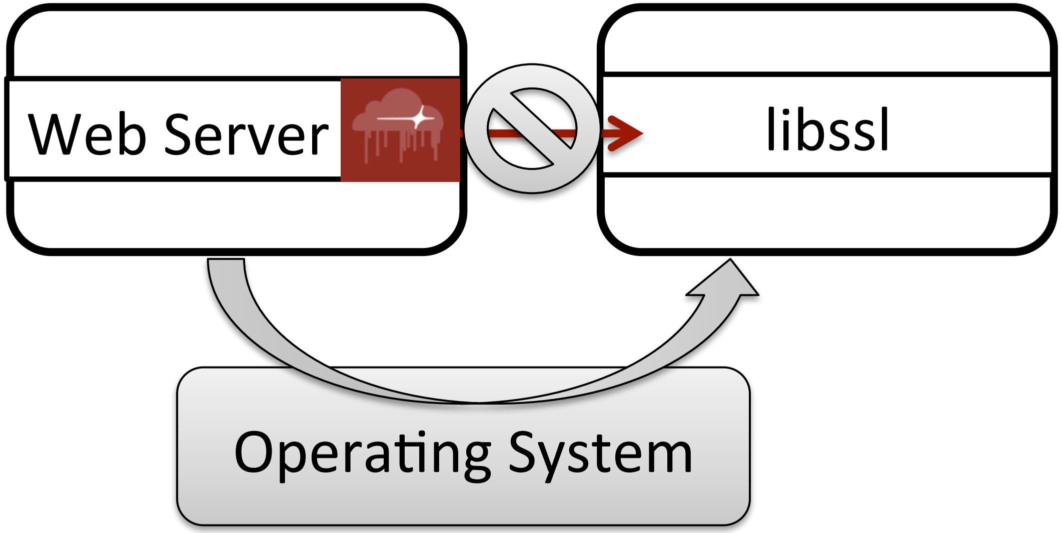
libevent

libcrypto

libssl

Web Server

Single Enclave Application



Web Server

libssl

Operating System

Multi-Enclave Application

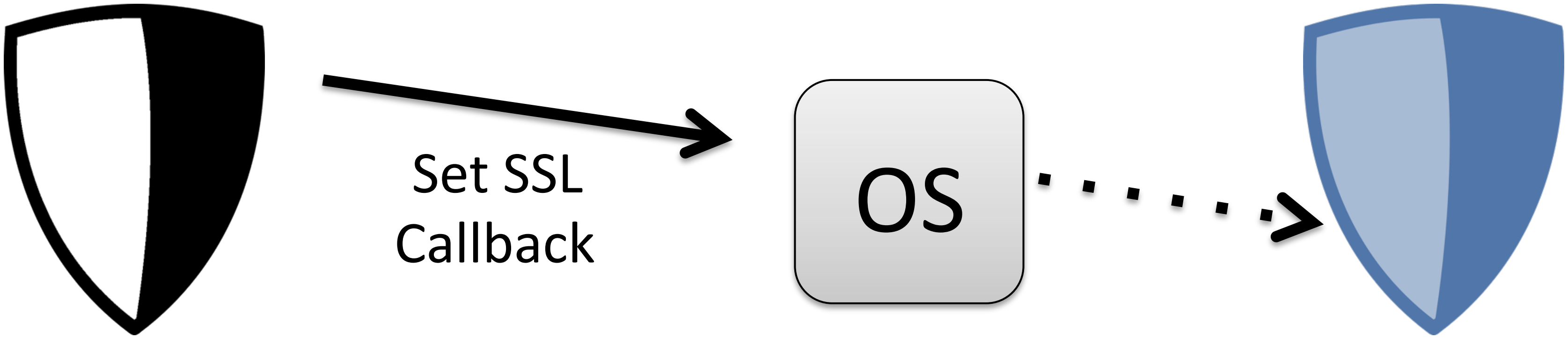
Attacks on Multi-Enclave Applications

```
session_t session;
certificate_credentials_t xcred;

/* Specify callback function*/
certificate_set_verify_function (...);

/* Initialize TLS session */
init (&session, TLS_CLIENT);
```

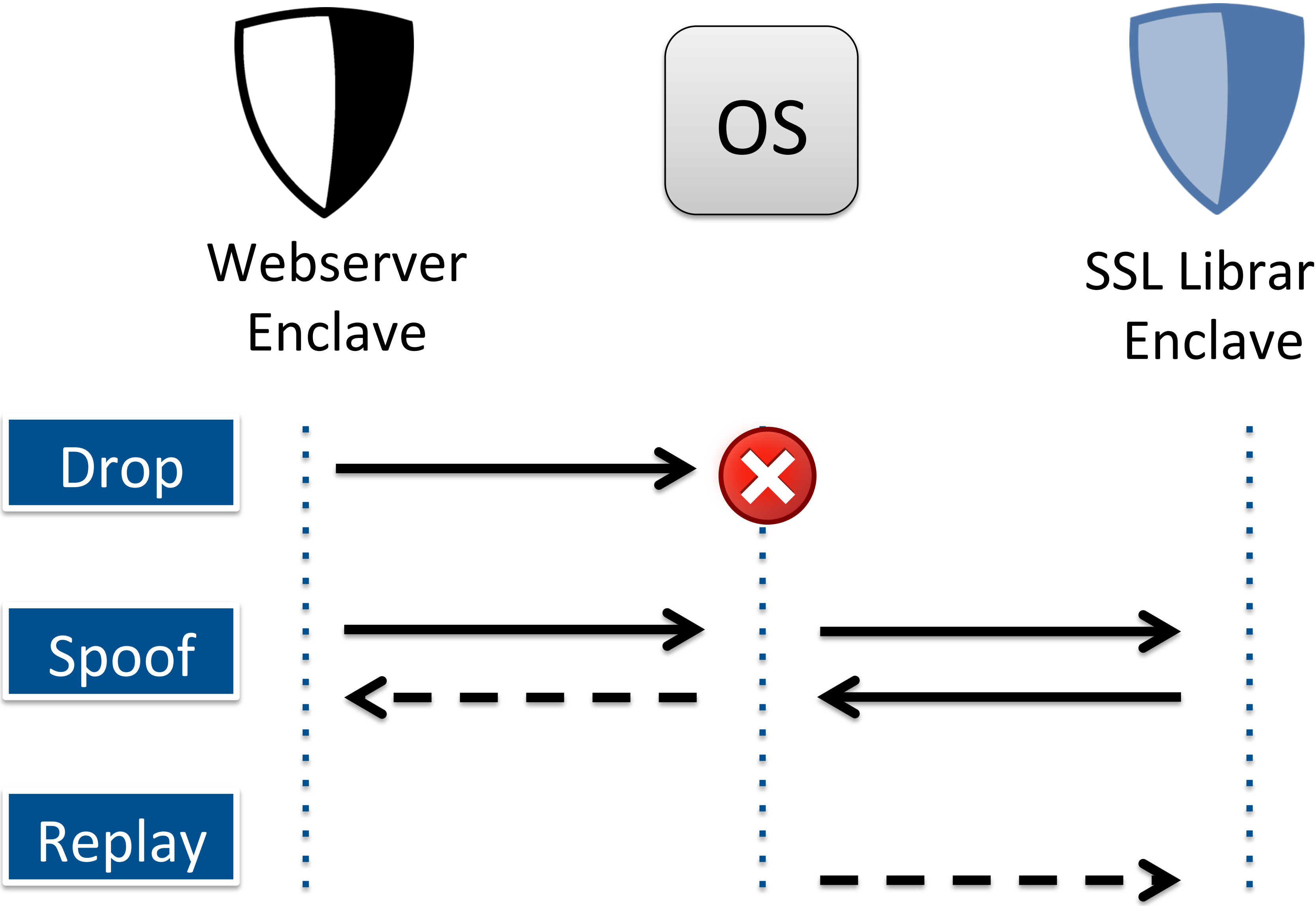
[SSL Lib]



Webserver
Enclave

SSL Library
Enclave

Attacks on Multi-Enclave Applications



Webserver Enclave

OS

SSL Library Enclave

Drop

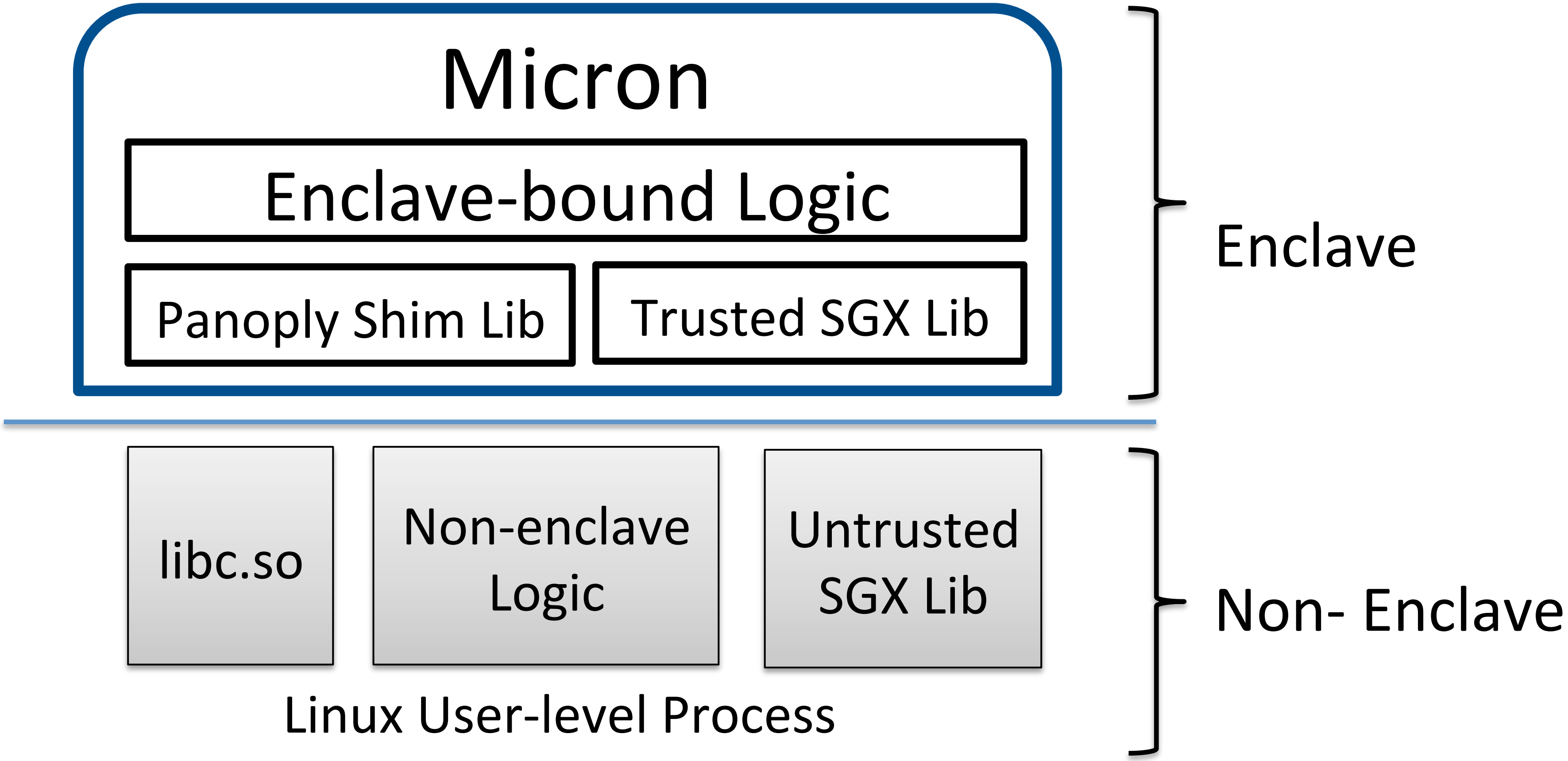
Spoof

Replay

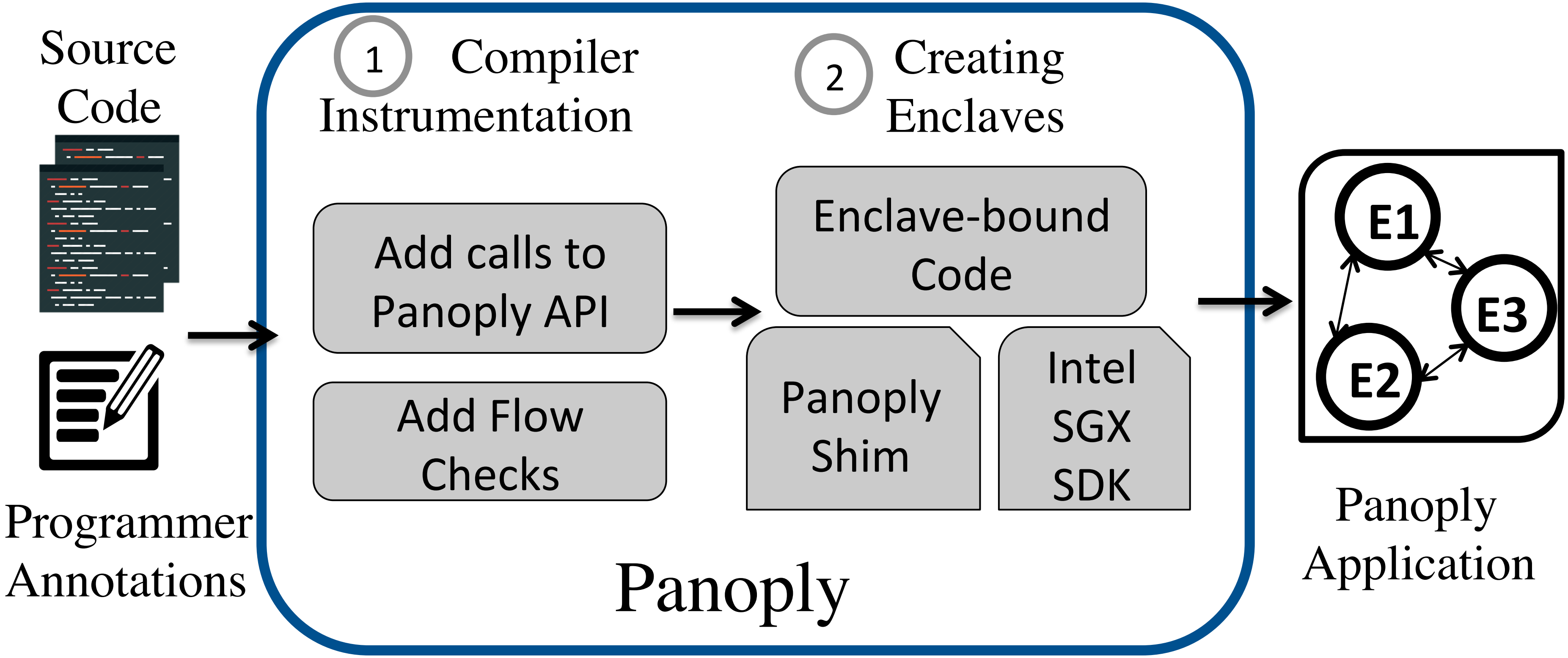
Our Solution: Panoply

Panoply Runtime

Microns keep libc outside the enclave

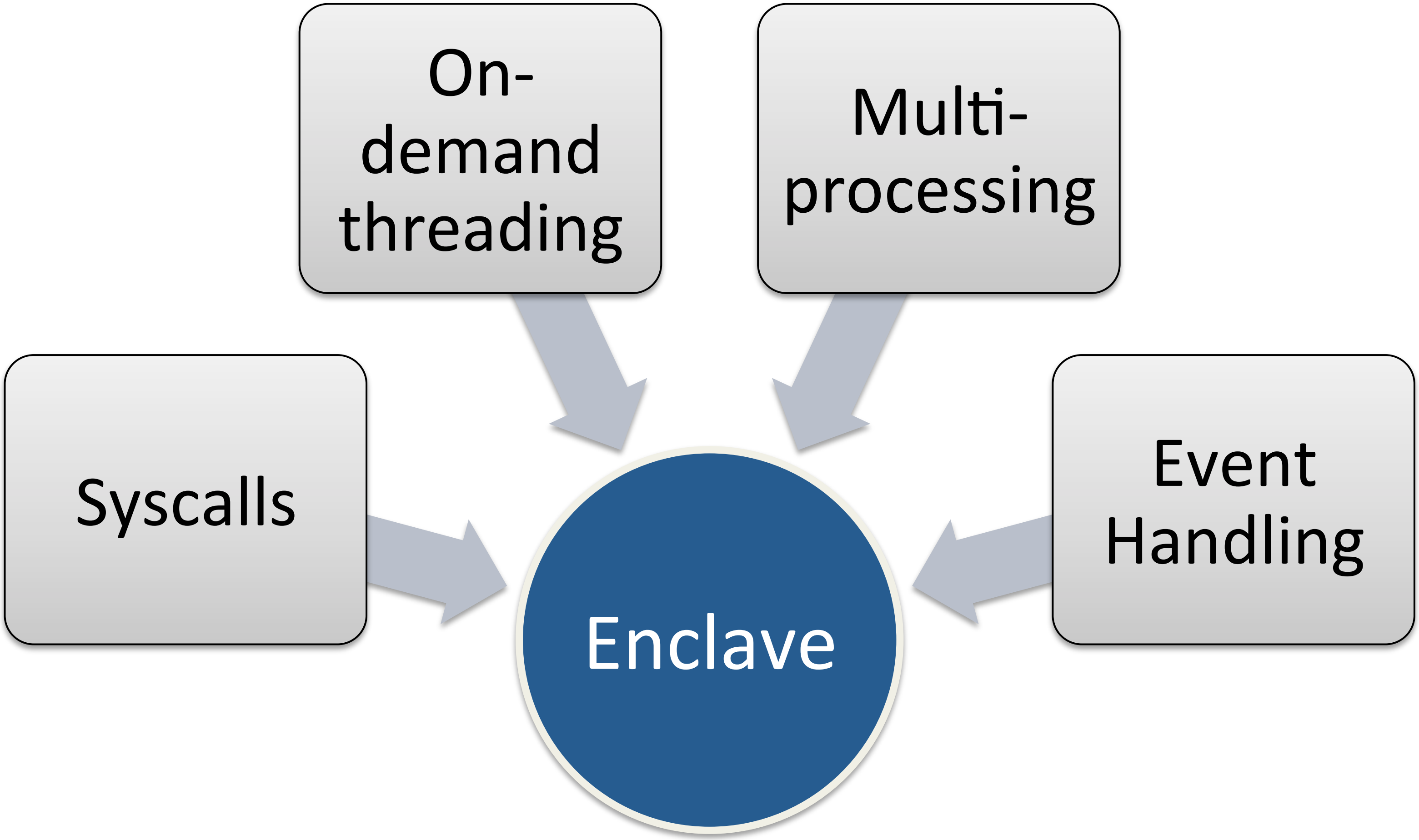


Overview


Source Code
Programmer Annotations
1 Compiler Instrumentation
Add calls to Panoply API
Add Flow Checks
2 Creating Enclaves
Enclave-bound Code
Panoply Shim
Intel SGX SDK
Panoply
E1
E3
E2
Panoply Application

Challenge I: Expressiveness

Delegate rather than emulate



On-demand threading

Multi-processing

Syscalls

Event Handling

Enclave

Expressiveness: Panoply APIs

Core Services

Process Creation and Control	5
Signals	6
Timers	5
File and Directory Operations	37
Pipes	4
C Library (Standard C)	66
I/O Port Interface and Control	40

Thread Extensions

Thread Creation, Control, and Cleanup	17
Thread Scheduling	4
Thread Synchronization	10
Signal Delivery	2
Signal Handling	3

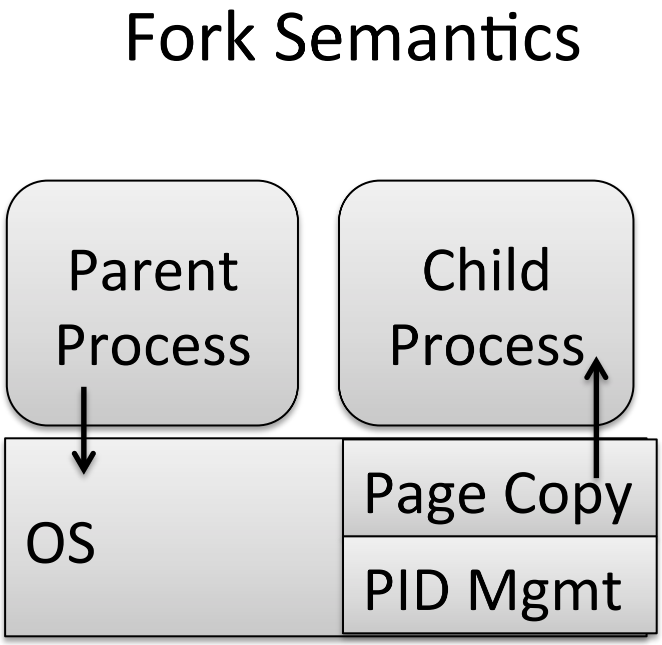
Real-time Extensions

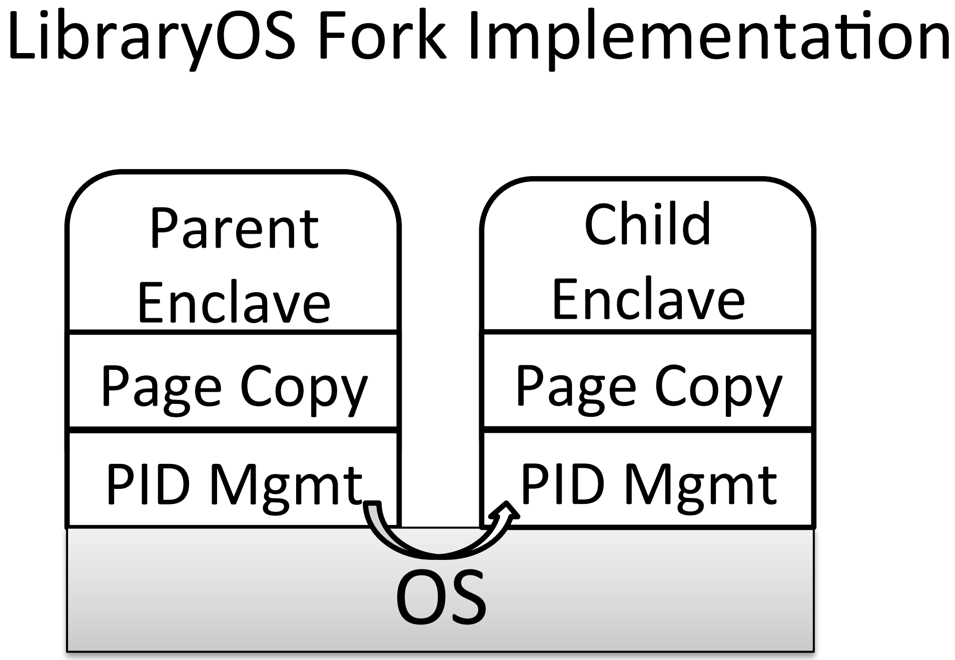
Real-Time Signals	4
Clocks and Timers	1
Semaphores	2
Message Passing	7
Shared Memory	6
Asynchronous and Synchronous I/O	29
Memory Locking Interface	6

POSIX APIs
Supported for
Commodity Linux Apps

Expressiveness Example: Fork

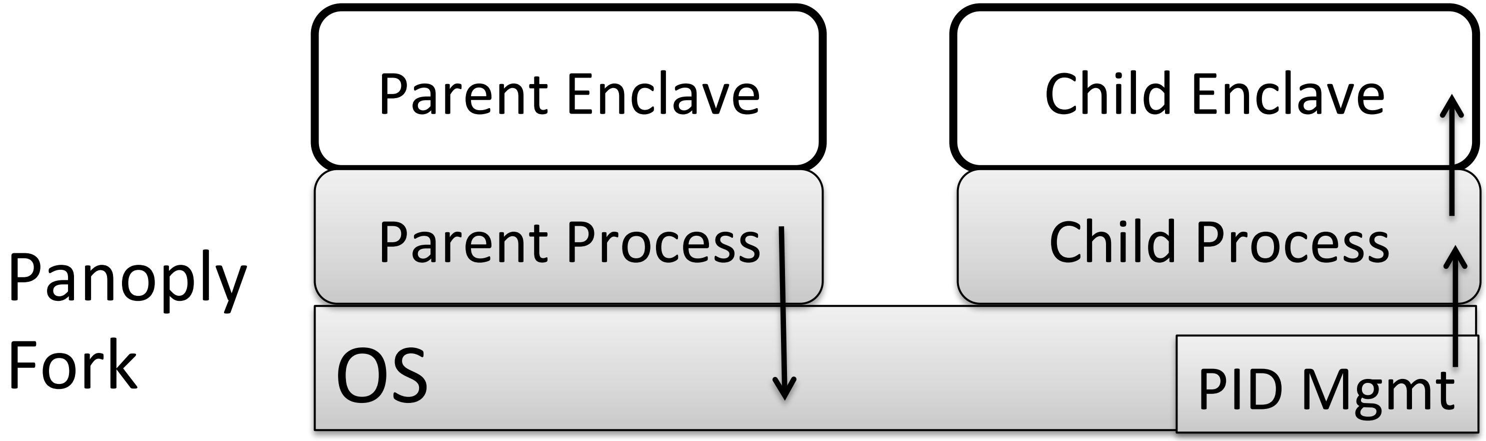
LibraryOSes emulate fork semantics





Expressiveness Example: Delegating Fork

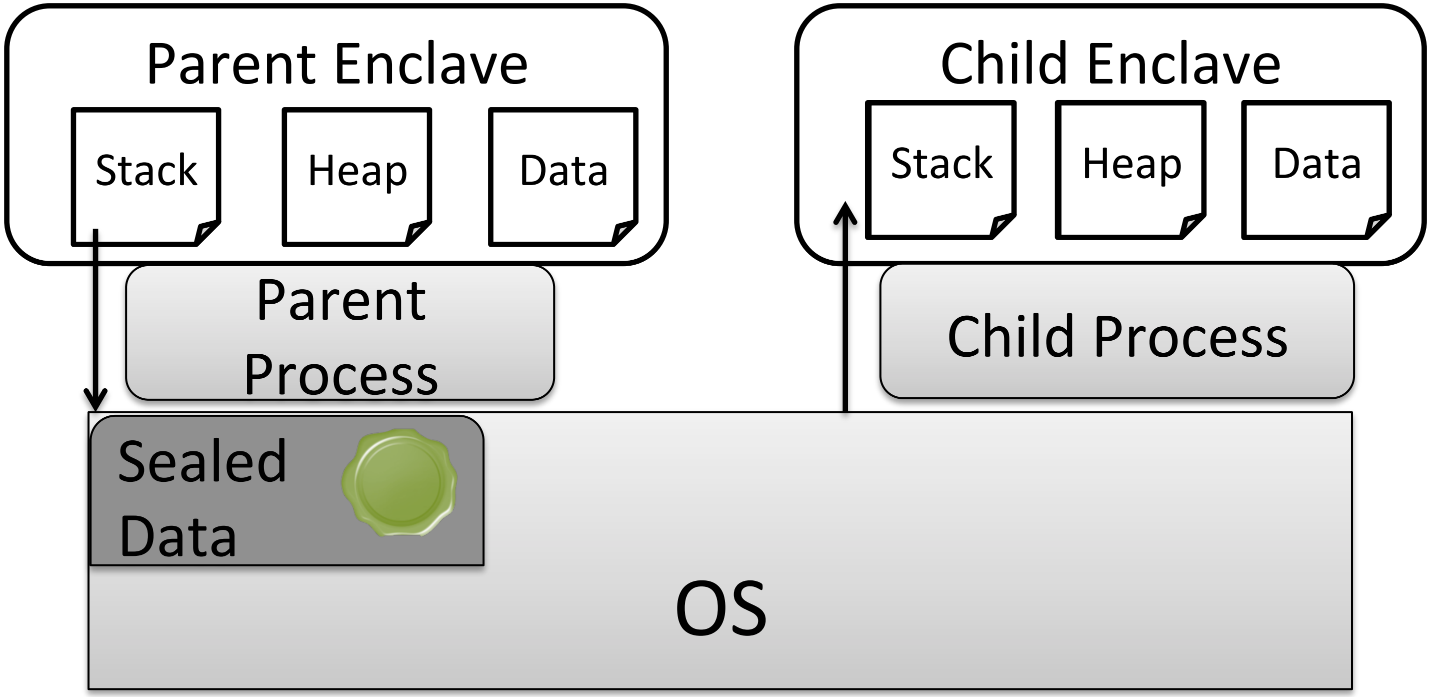
- Creating child process and child enclave



- Child enclave has a clean memory state

Expressiveness Example: Achieving Fork Semantics

- Mirroring parent's memory in child enclave
  - After the fork call, before resuming execution



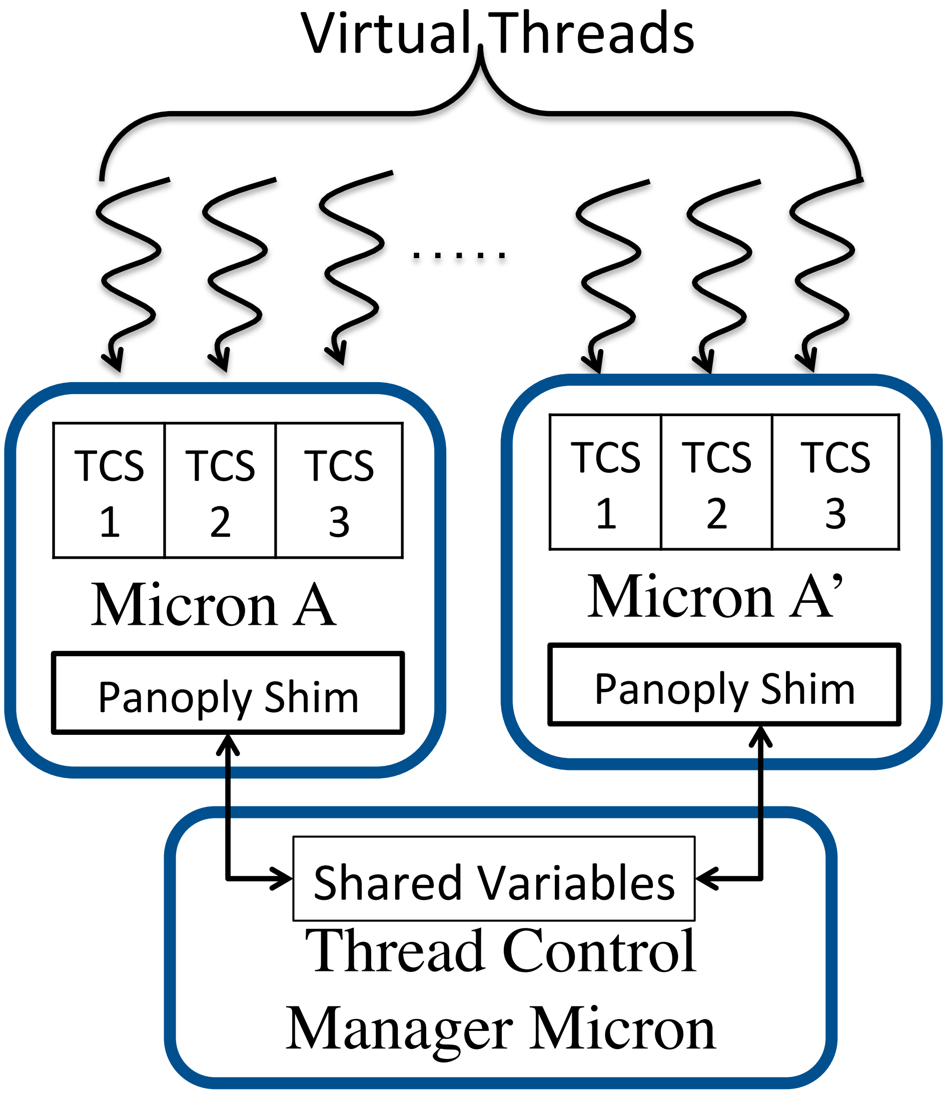
Expressiveness Example: Achieving Fork Semantics

- Mirroring parent's memory in child enclave
  - Full replica: default mode in Panoply

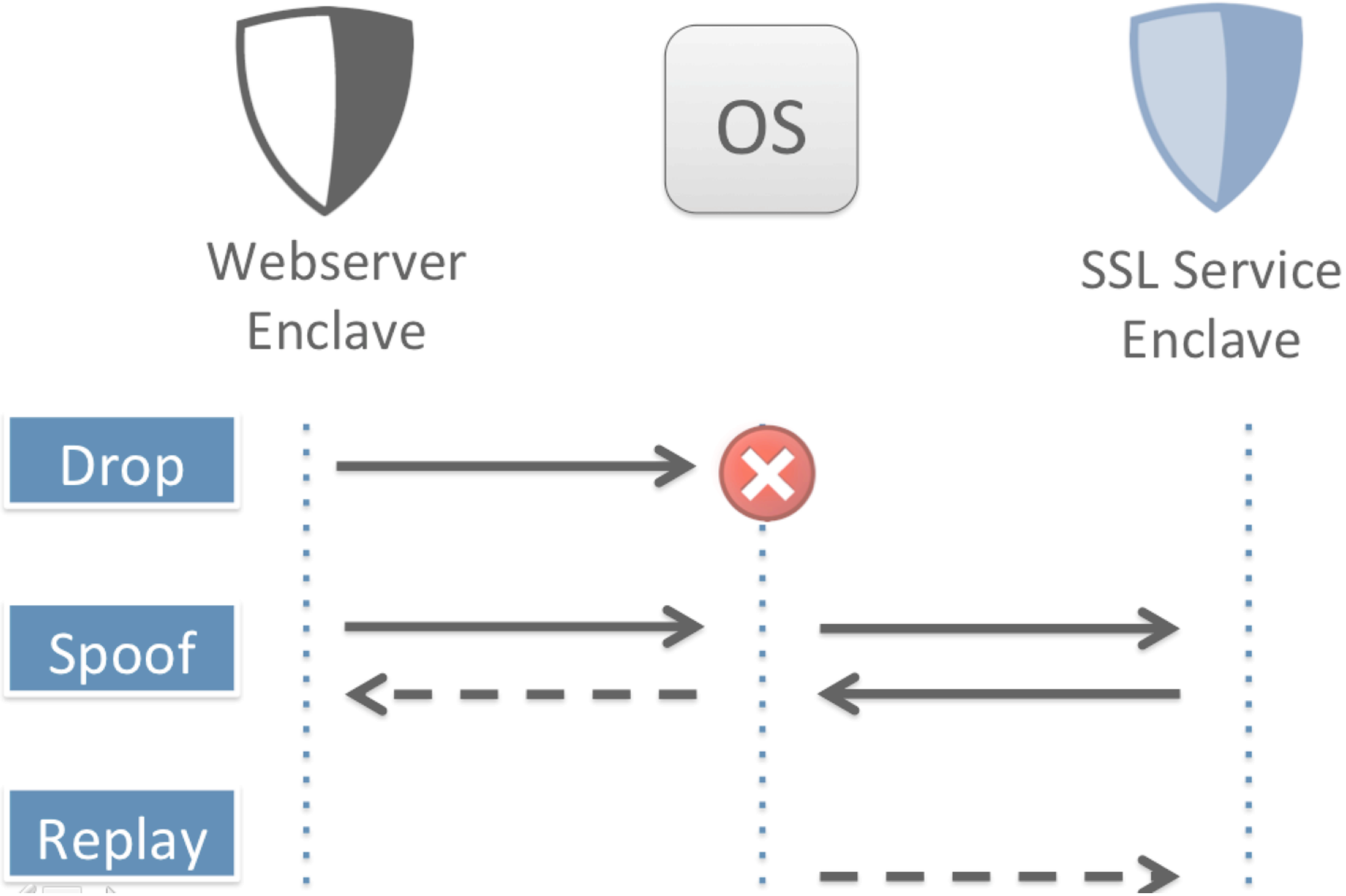
- Alternative strategies to full replica
  - Copy on demand: Requires page-fault support from SGX v2

  - Copy on need: Replicate selected addresses which are pre-determined by static analysis

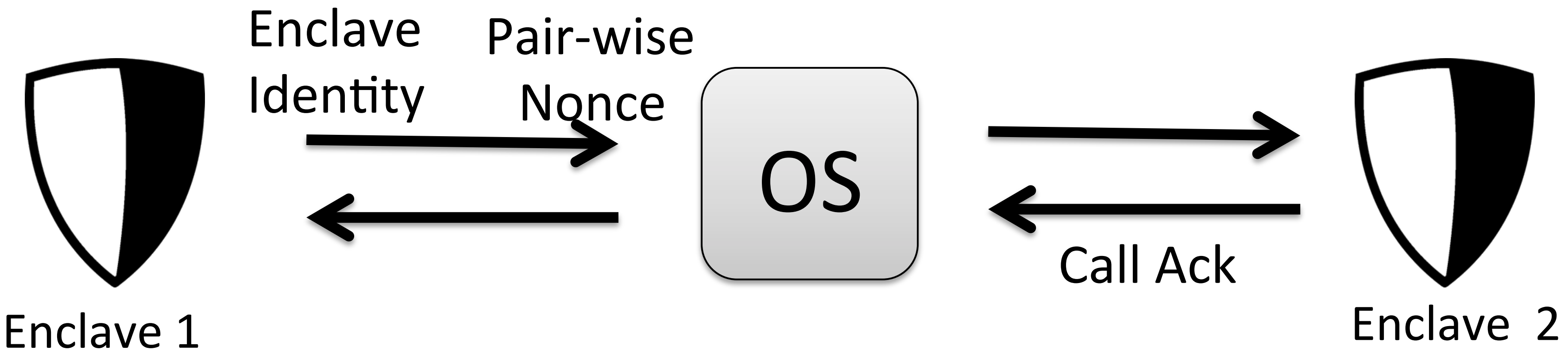
Expressiveness Example: Multi-Threading



Challenge II: Multi-enclave Applications



Securing Multi-Enclave Apps



Attack	Security Property
Spoofing	Sender / Receiver Authentication
Replay	Message Freshness
Silent Drops	Reliable Delivery

Evaluation

Benchmarks

- Real-world use-cases for SGX
  - **4** apps: Tor, H2O web server, FreeTDS, OpenSSL
- Operating system stress testing
  - **26** LMBench benchmarks tests
  - **17** metrics for memory, network, signal, syscall APIs

TCB Evaluation

Panoply

Component	LOC
Panoply Library	10425
API Wrappers	9788
Total	20213

Graphene-SGX

Component	LOC
Glibc	1156740
libPal-LinuxSGX	16901
libPal-enclave	33103
Total	1206744

Panoply reduces TCB by 2 orders of magnitude

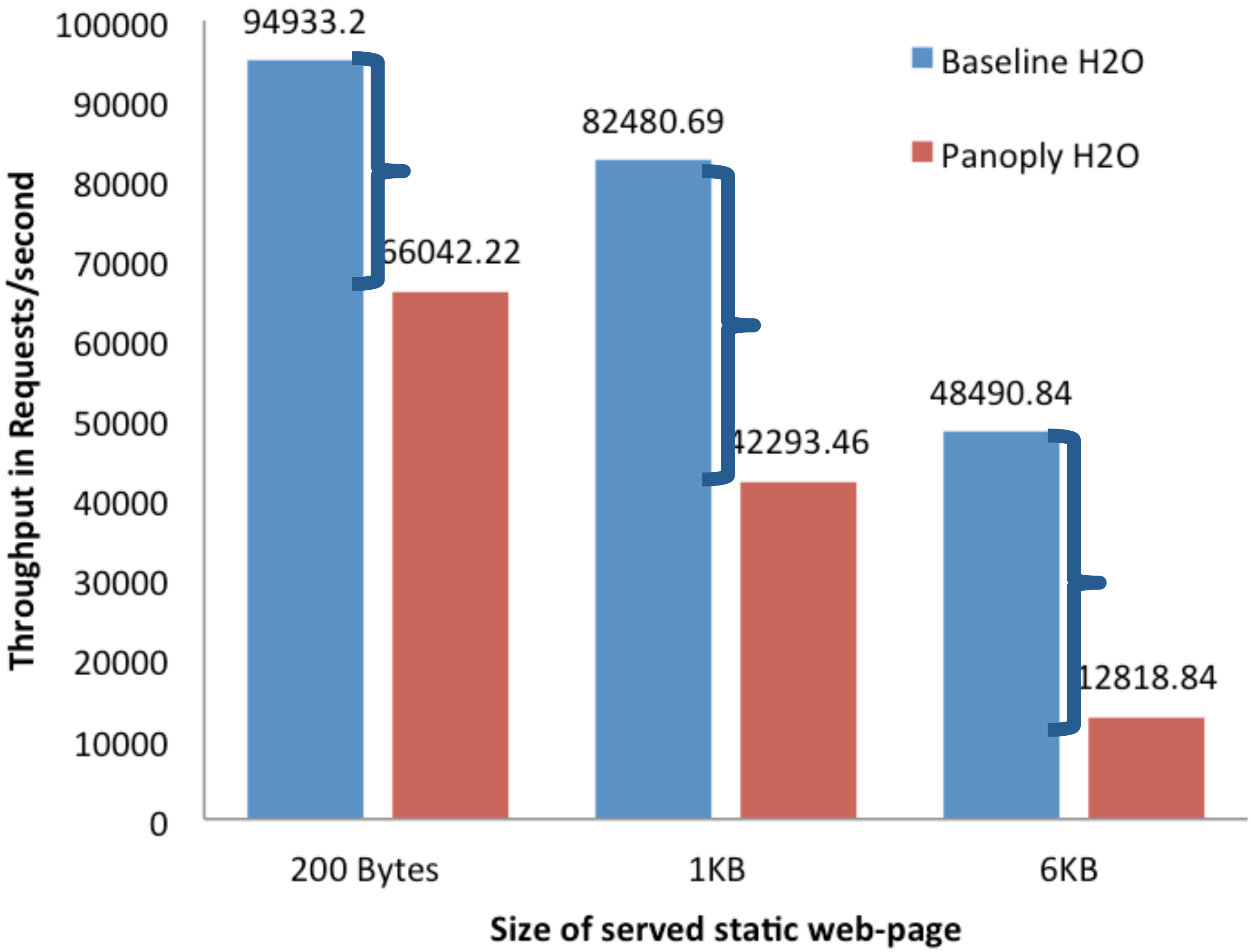
Performance Evaluation

- Create delete takes large fraction of the time
- Overhead increases with number of Out-Calls

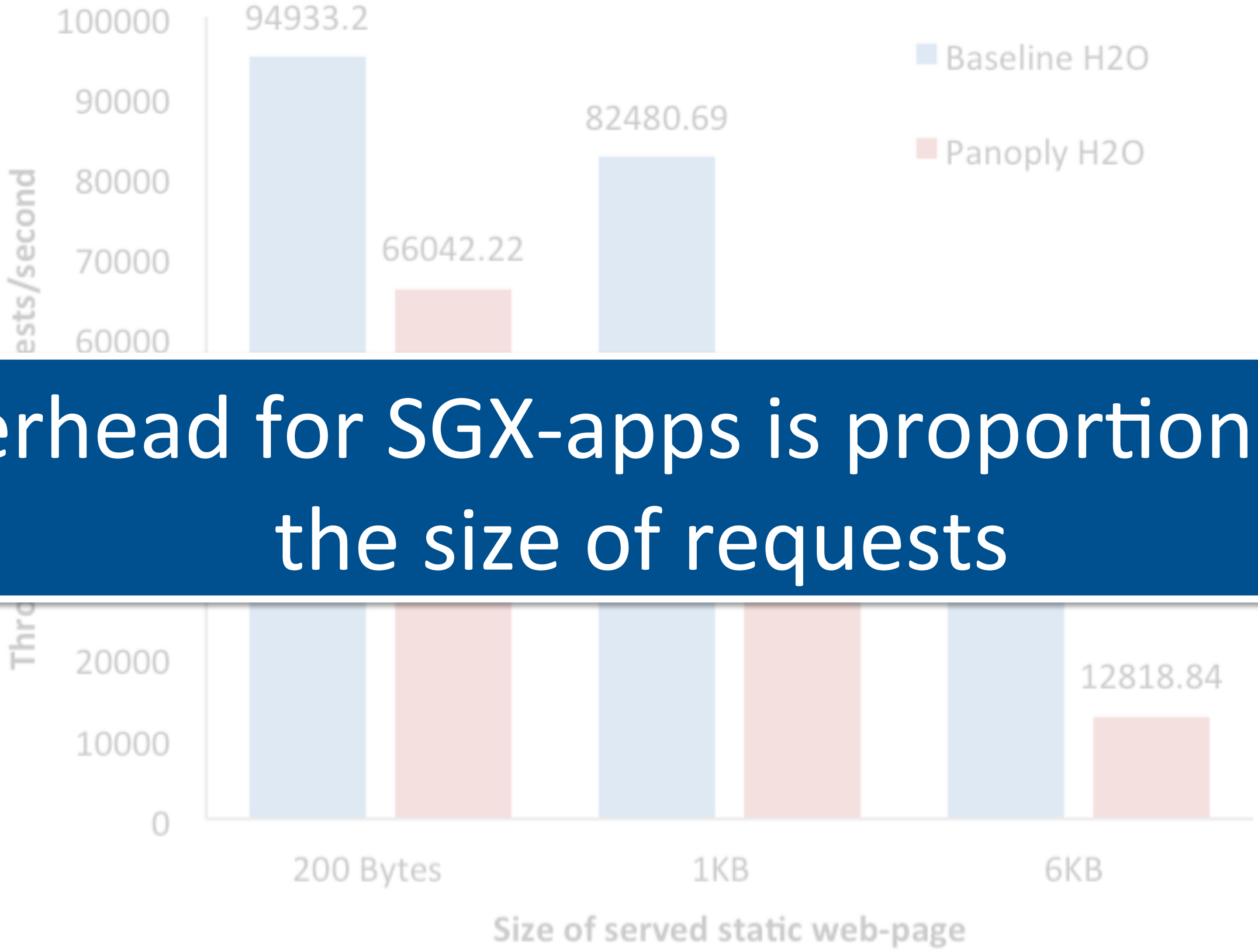
Panoply incurs 24% overhead

App	Panoply	Empty Enclave	Overhead (% increase)
OpenSSL	3.16	2.79	**13**
H2O	8.79	6.56	**34**
FreeTDS	8.74	8.60	**1**
Tor	6.72	4.54	**48**
		Average	**24**

Throughput Evaluation

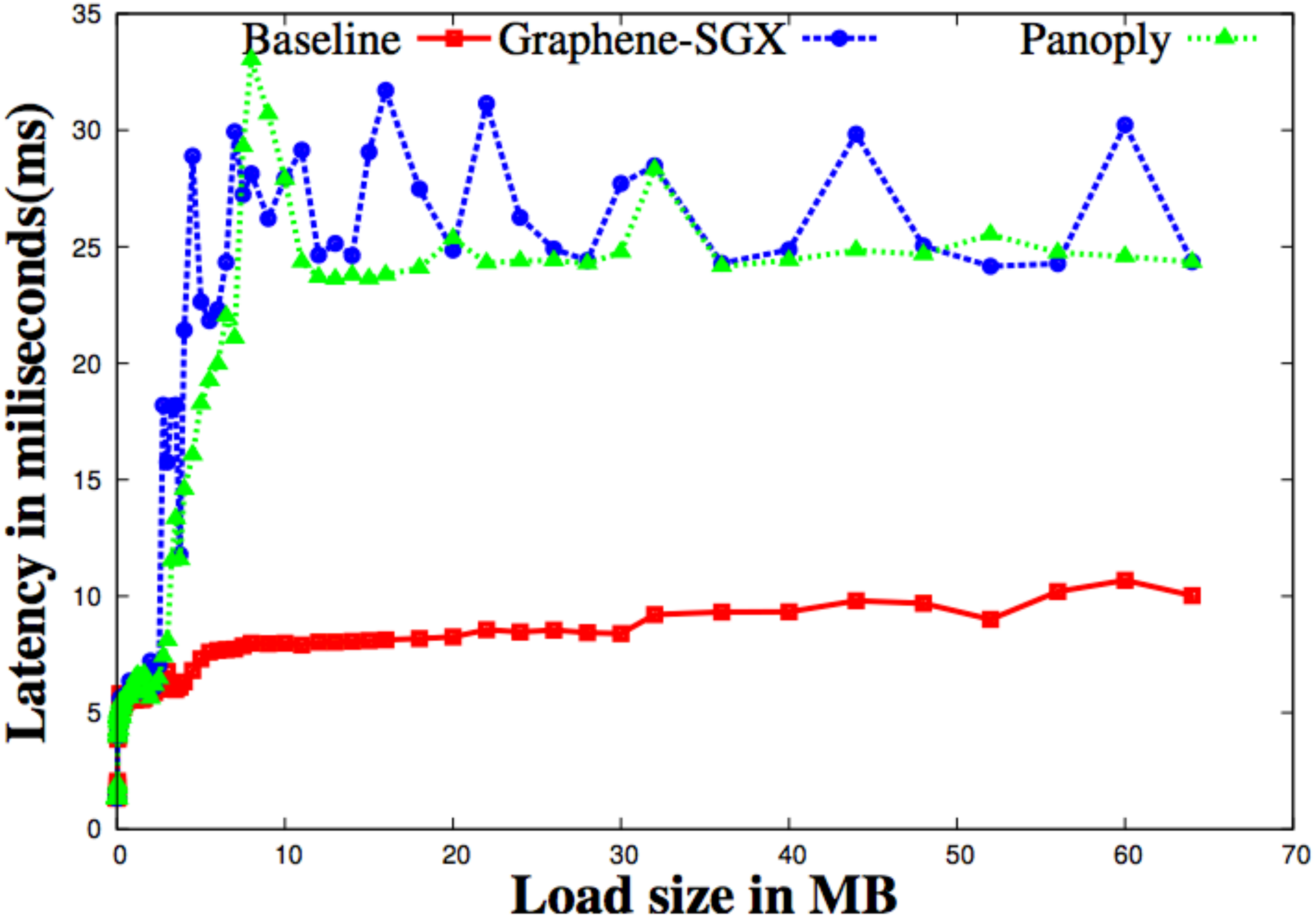


Throughput Evaluation

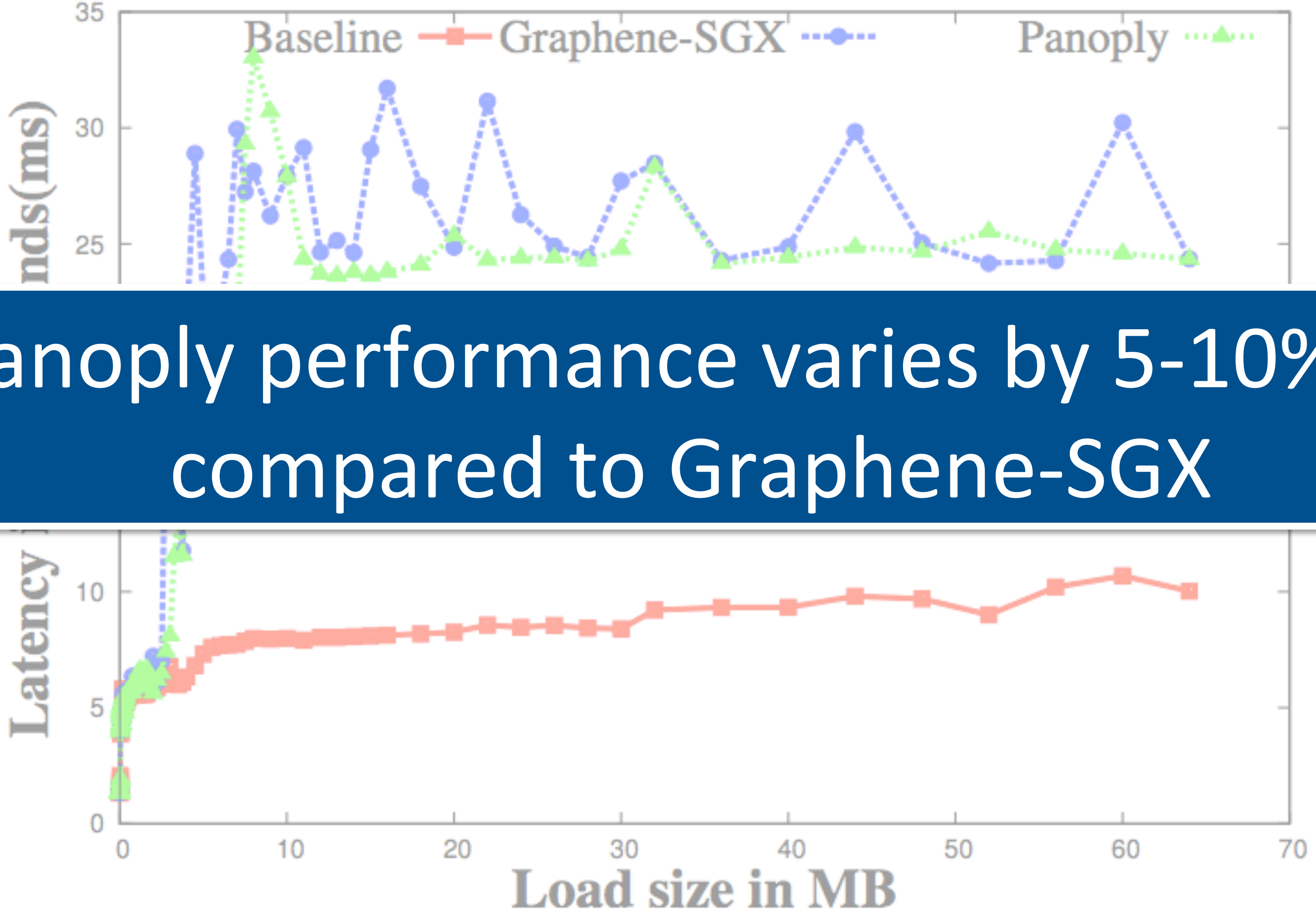


Overhead for SGX-apps is proportional to the size of requests

Comparison with Graphene-SGX

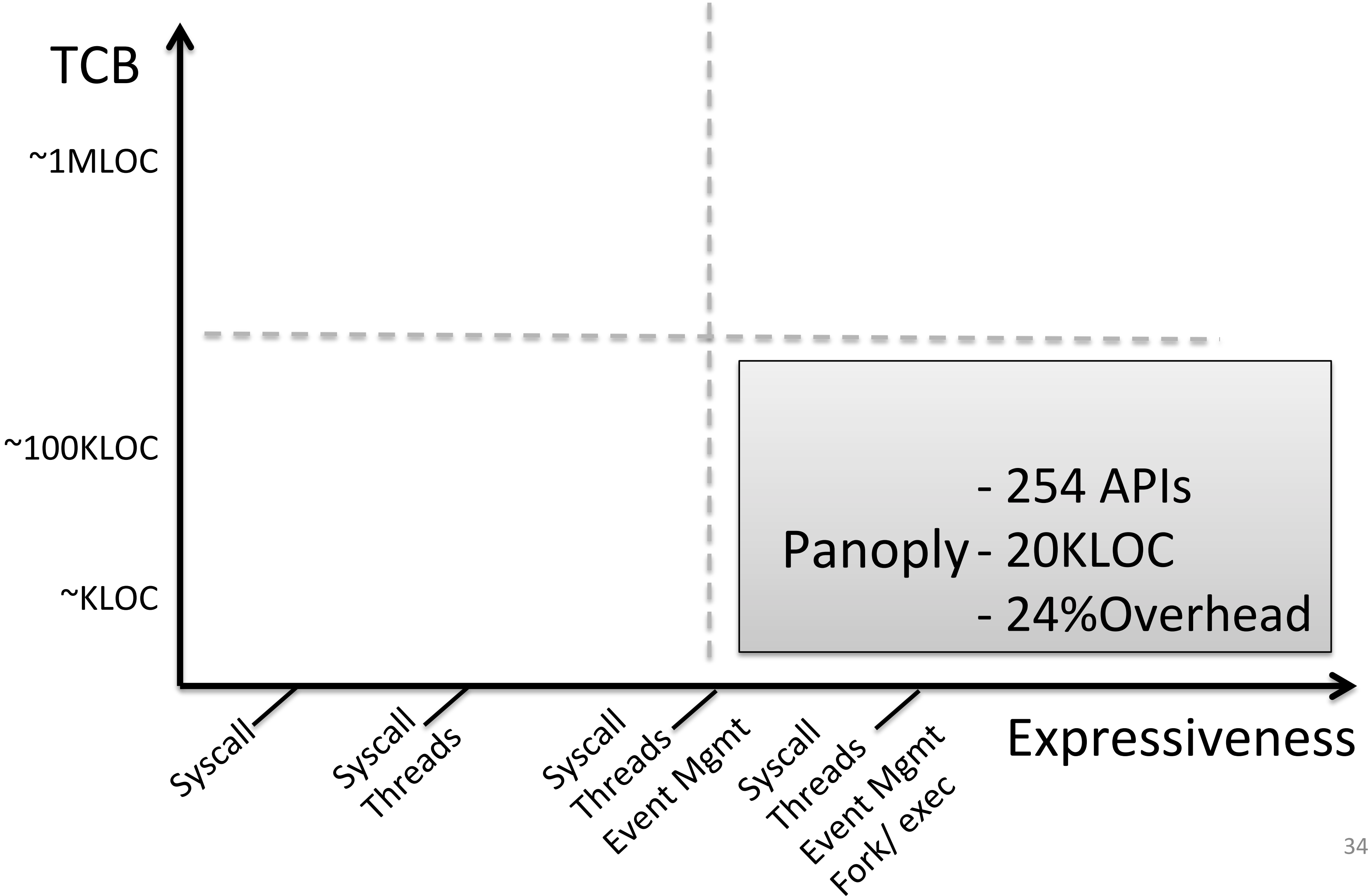


Comparison with Graphene-SGX



Panoply performance varies by 5-10% as compared to Graphene-SGX

Conclusion



Panoply
- 254 APIs
- 20KLOC
- 24%Overhead

Contact

- Shweta Shinde

  shweta24@comp.nus.edu.sg

- Panoply Benchmarks & Case-studies: http://shwetasshinde24.github.io/Panoply/

Thank You !

References

- **[OSDI’ 14]** A. Baumann, M. Peinado, and G. Hunt, Shielding Applications from an Untrusted Cloud with Haven
- **[OSDI’ 16]** S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O’Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer, SCONE: Secure Linux Containers with Intel SGX
- **[OSDI’ 16]** T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data
- **[EuroSys’ 14]** Graphene-SGX Library OS - a library OS for Linux multi-process applications, with Intel SGX support