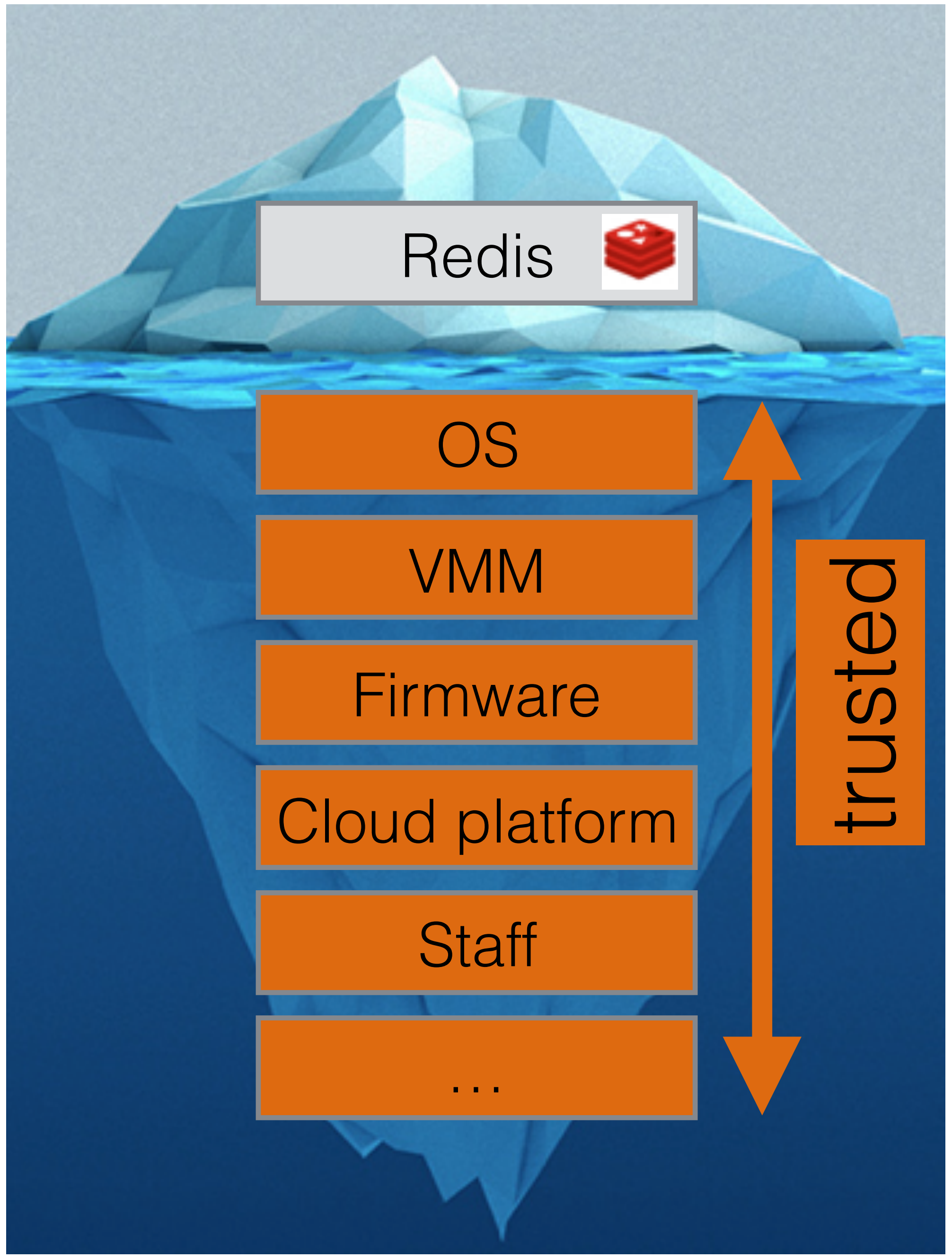
SCONE: **S**ecure Linux **Con**tainer **E**nvironments with Intel SGX

S. Arnautov, B. Trach, F. Gregor, **Thomas Knauth**, and A. Martin, Technische Universität Dresden; C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, and M. Stillwell, Imperial College London; D. Goltzsche, Technische Universität Braunschweig; D. Eyers, University of Otago; R. Kapitza, Technische Universität Braunschweig; P. Pietzuch, Imperial College London; C. Fetzer, Technische Universität Dresden

thomas.knauth@tu-dresden.de

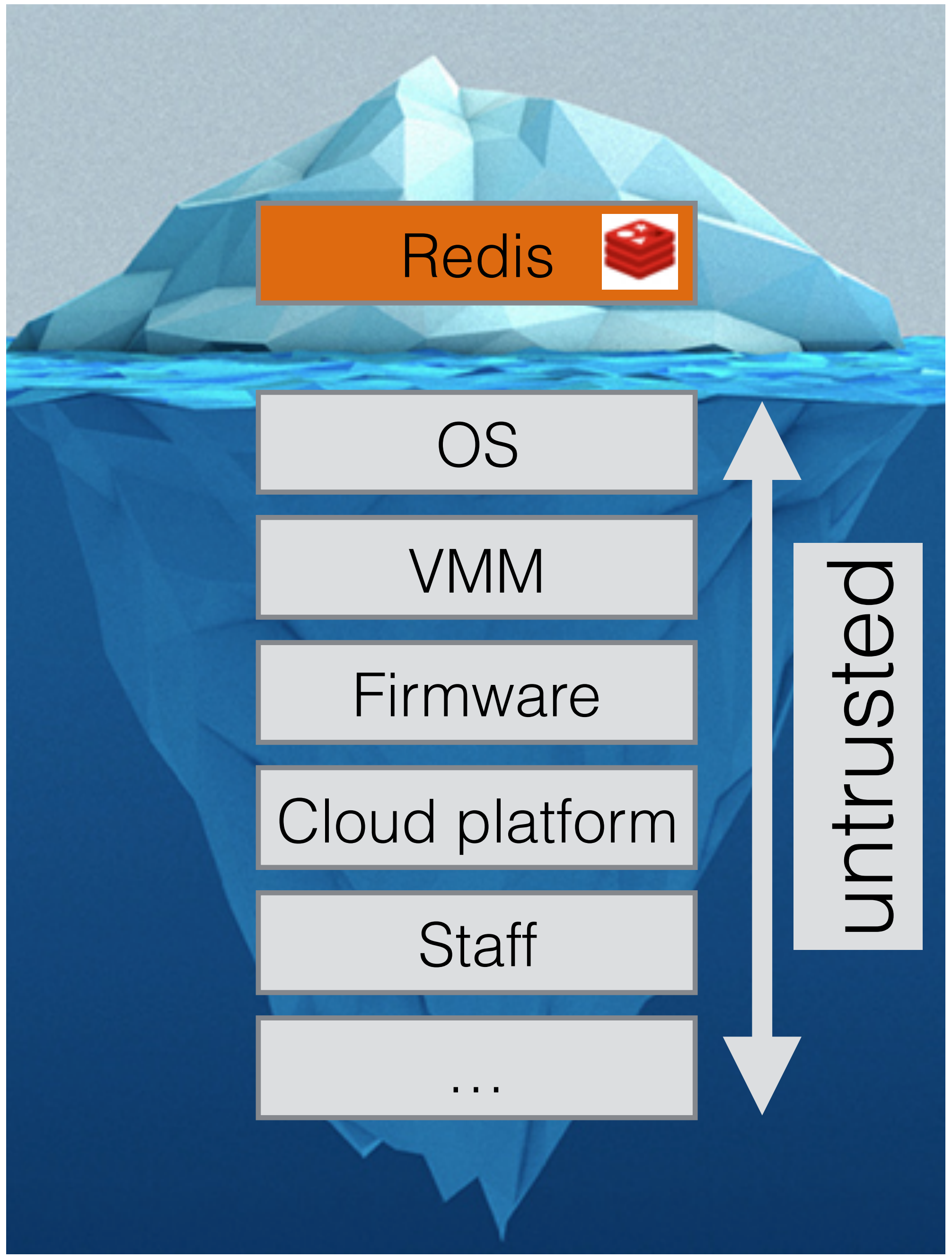
Trust Issues: The Provider's Perspective

- Cloud provider does not trust users
- Use virtual machines to isolate users from each other and the host
- VMs only provide one way protection



Trust Issues: The User's Perspective

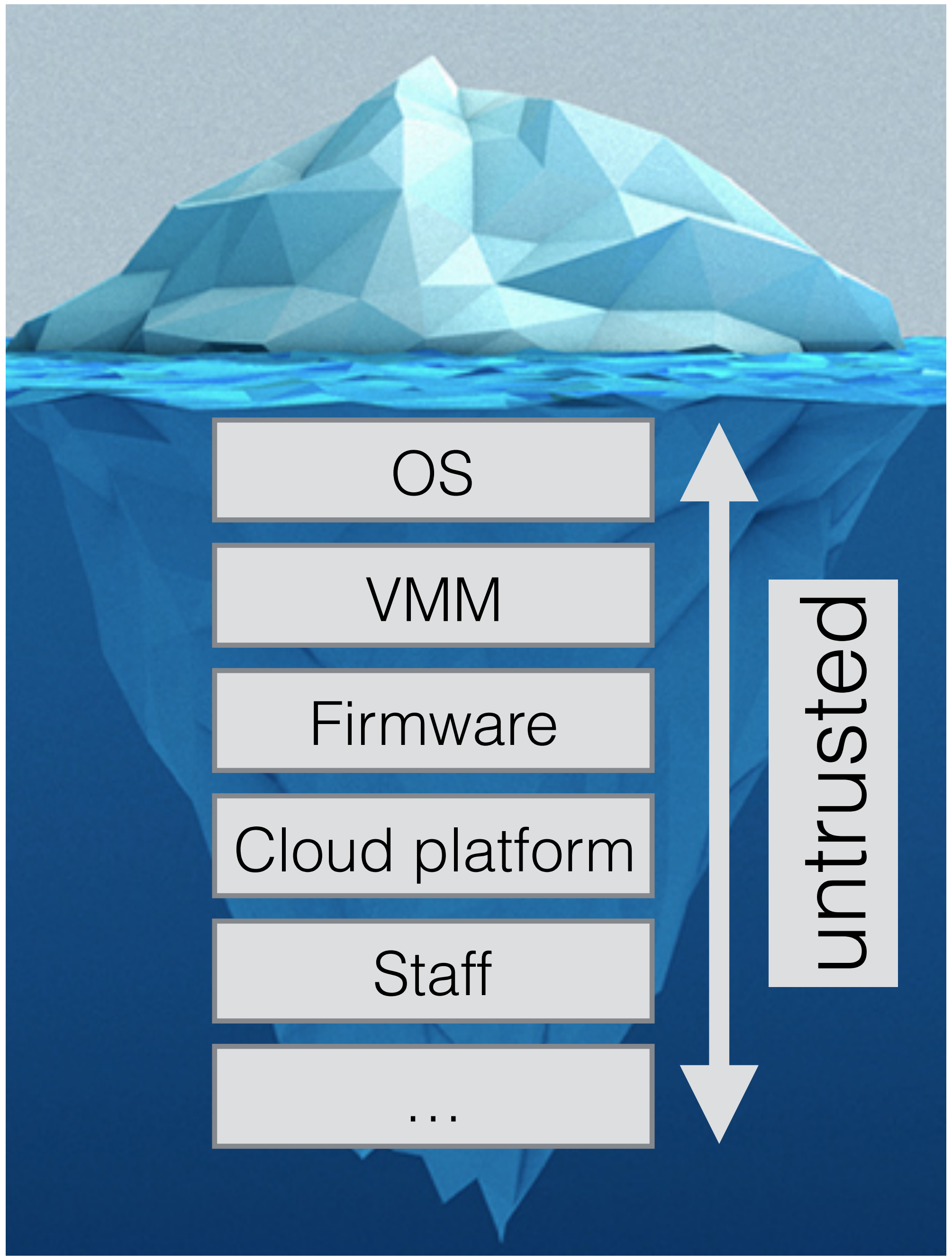
- Users trust their application
- Users must implicitly trust the cloud provider
- Existing applications implicitly assume trusted operating system

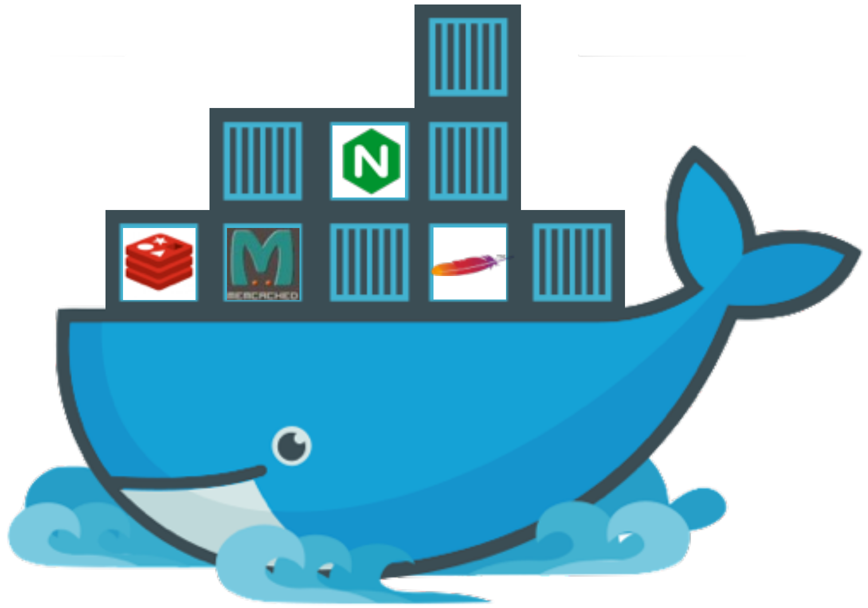


Containers are the new VMs

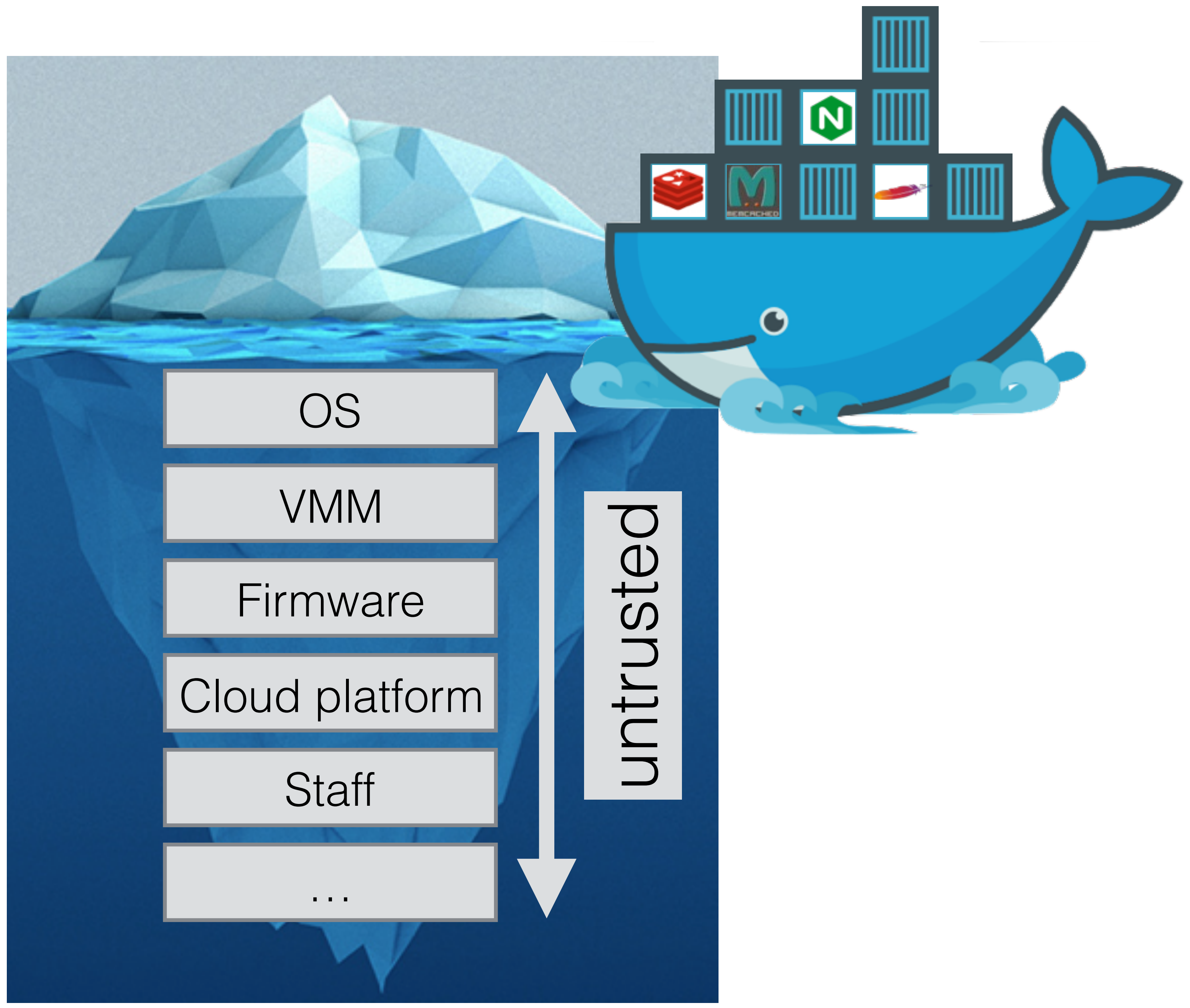
- Containers provide resource isolation and bundling
- Smaller resource overhead than virtual machines
- Convenient tooling to create and deploy applications in the cloud

Disaster!

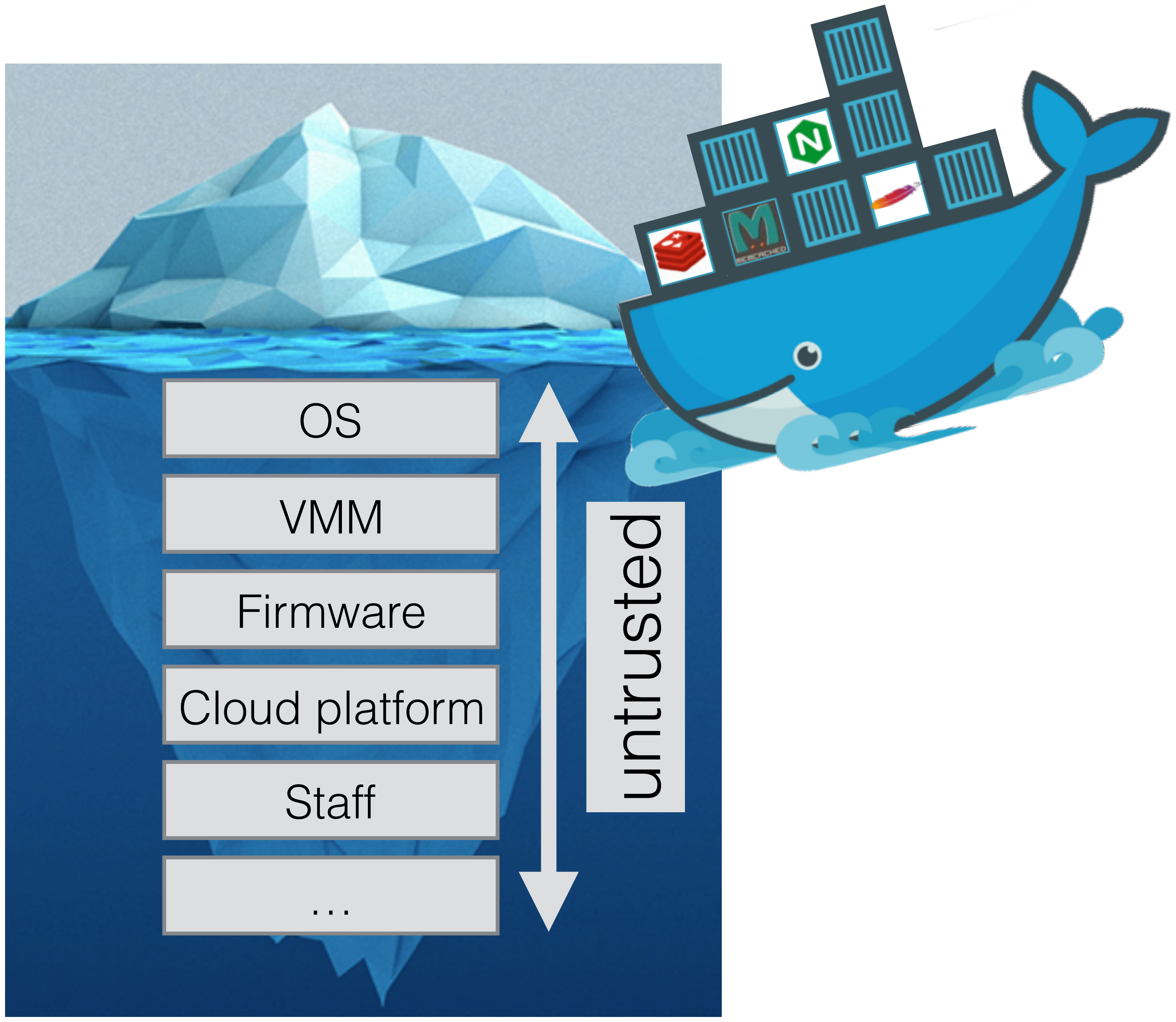




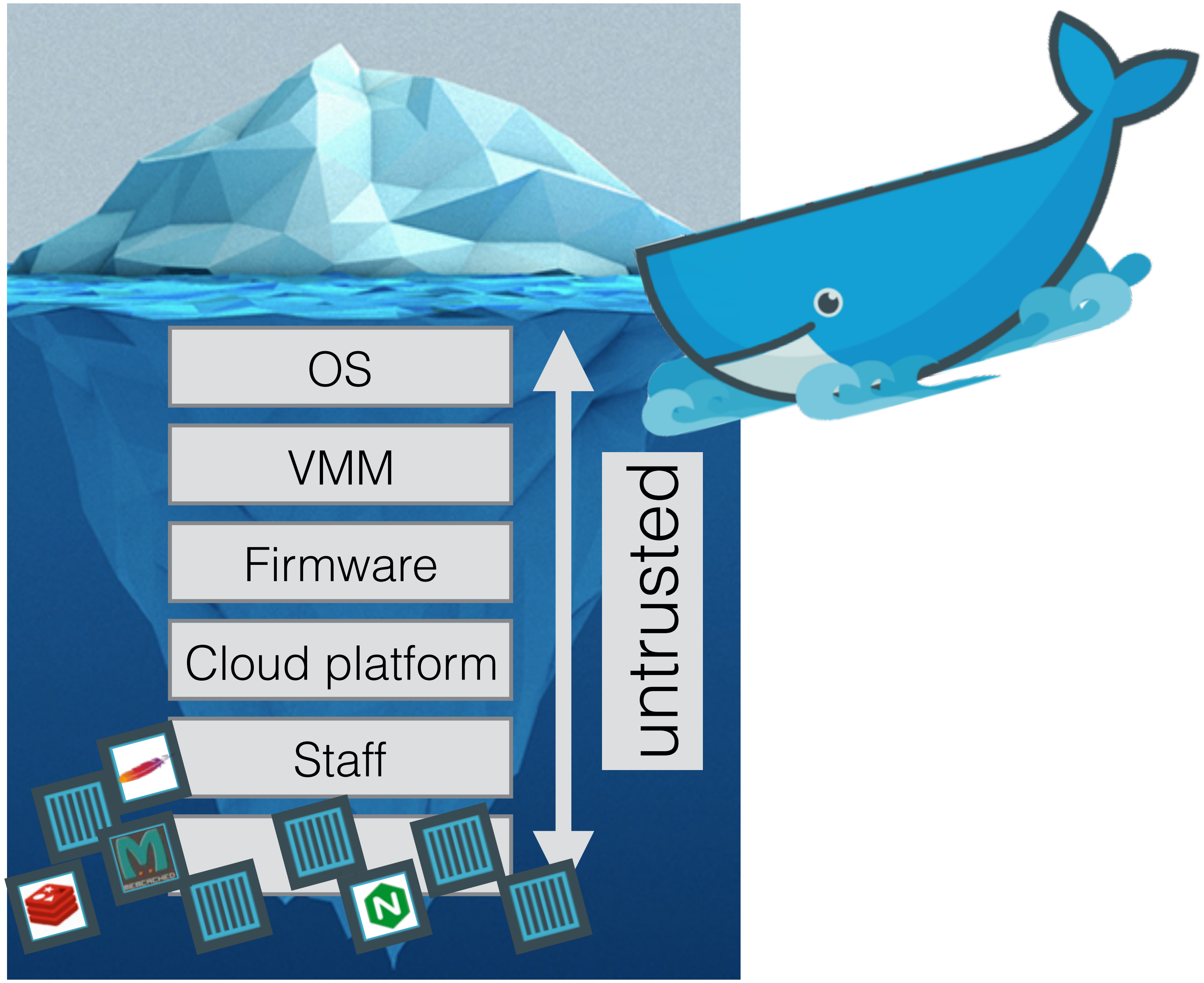
Disaster!



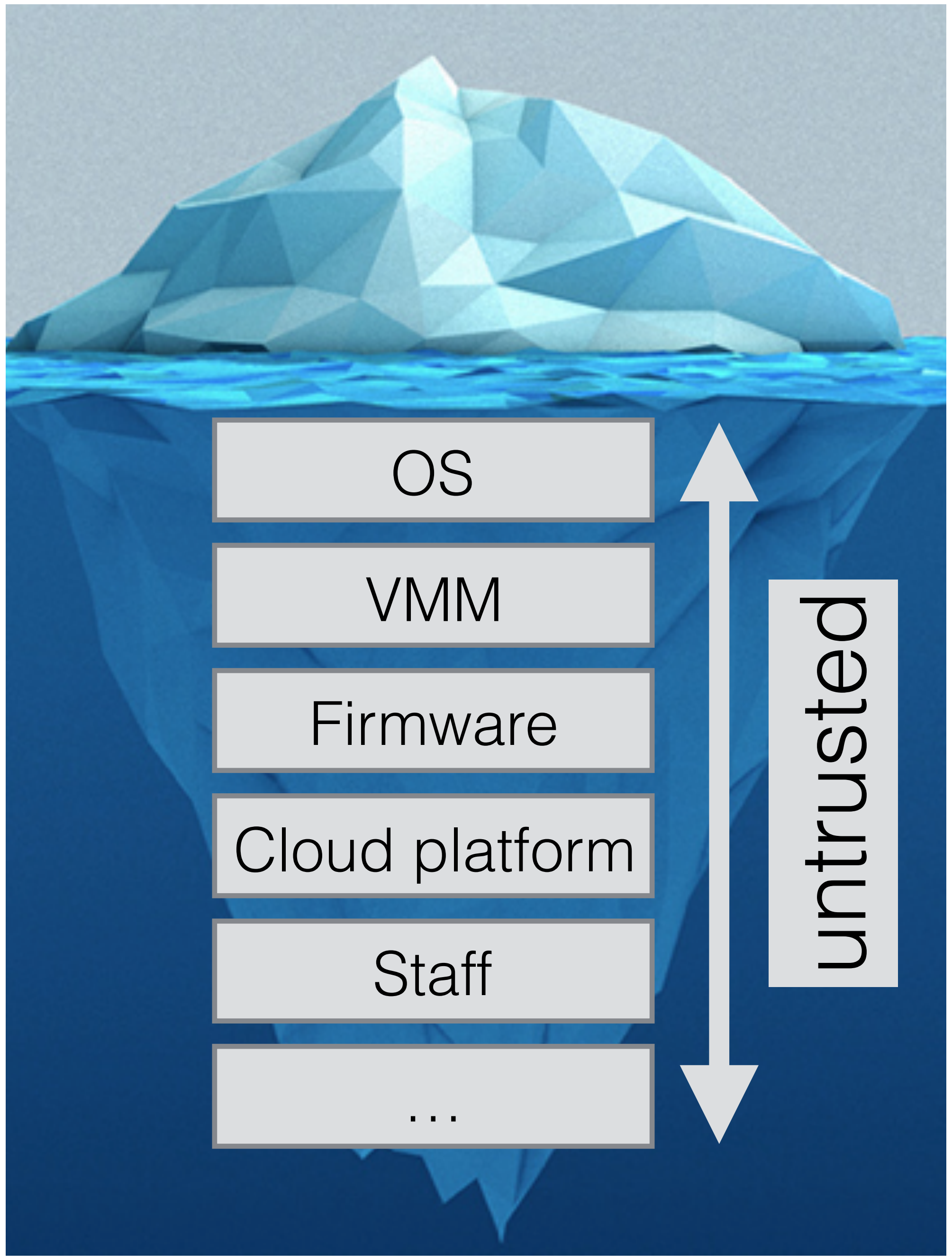
Disaster!



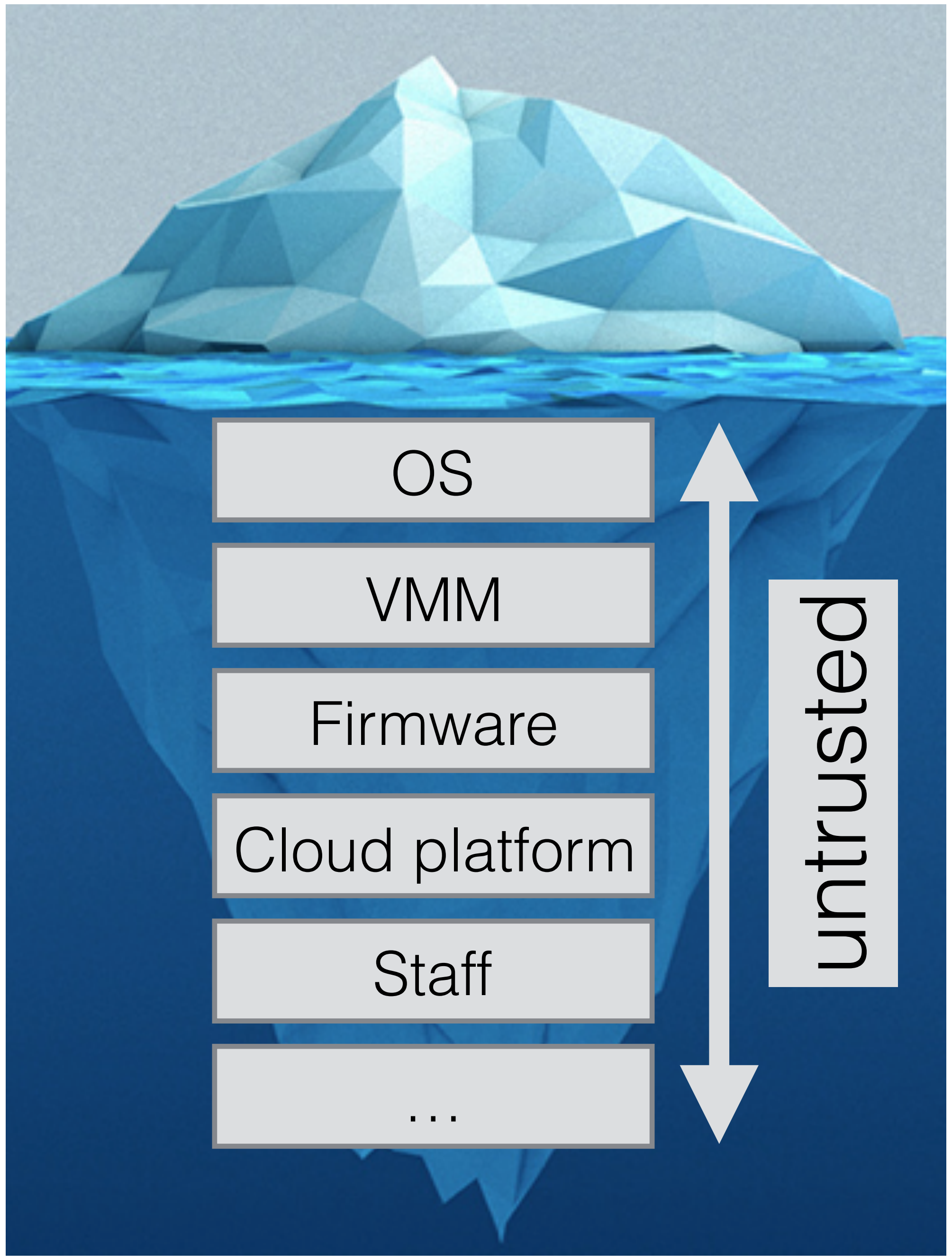
Disaster!



We want to …

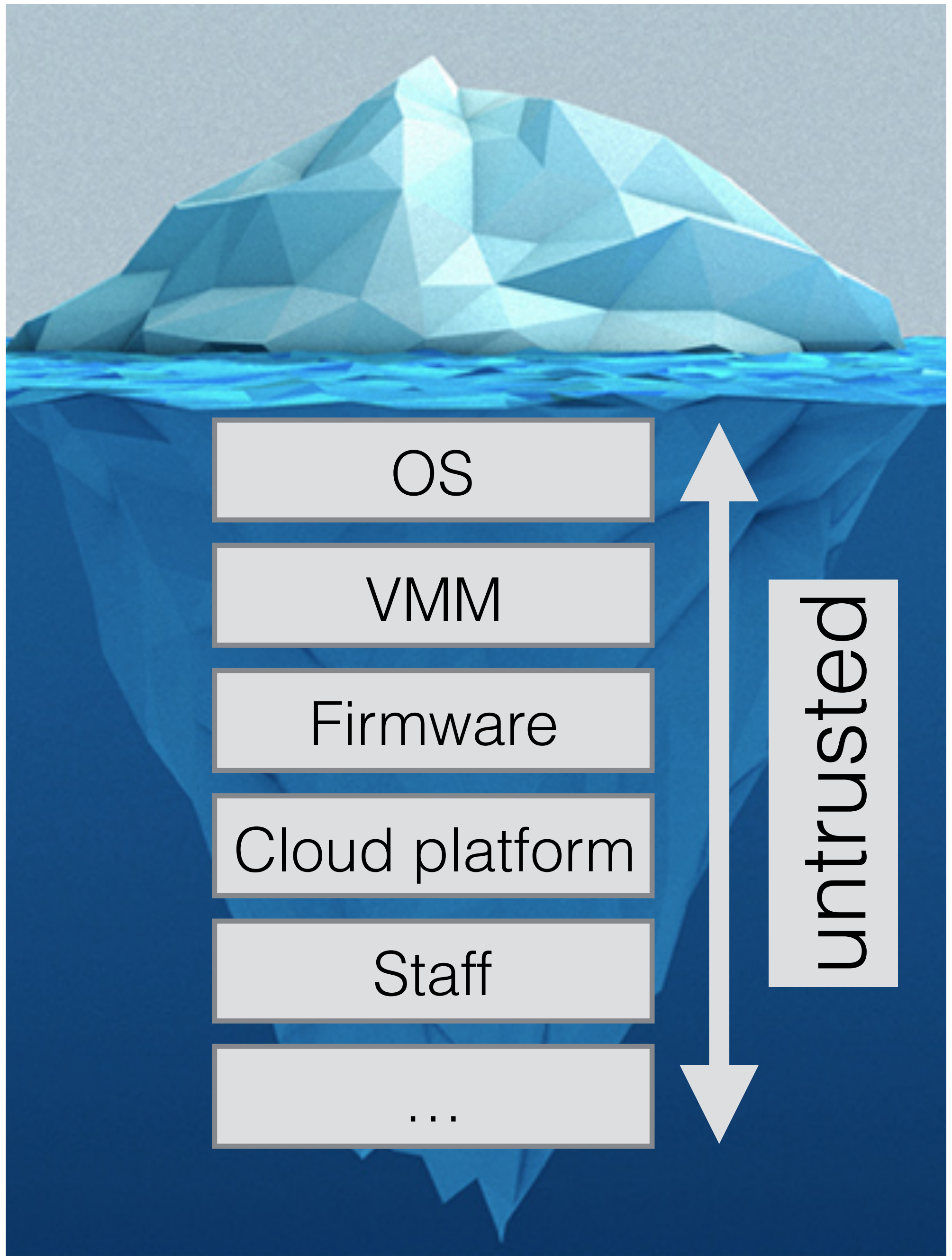


We want to …



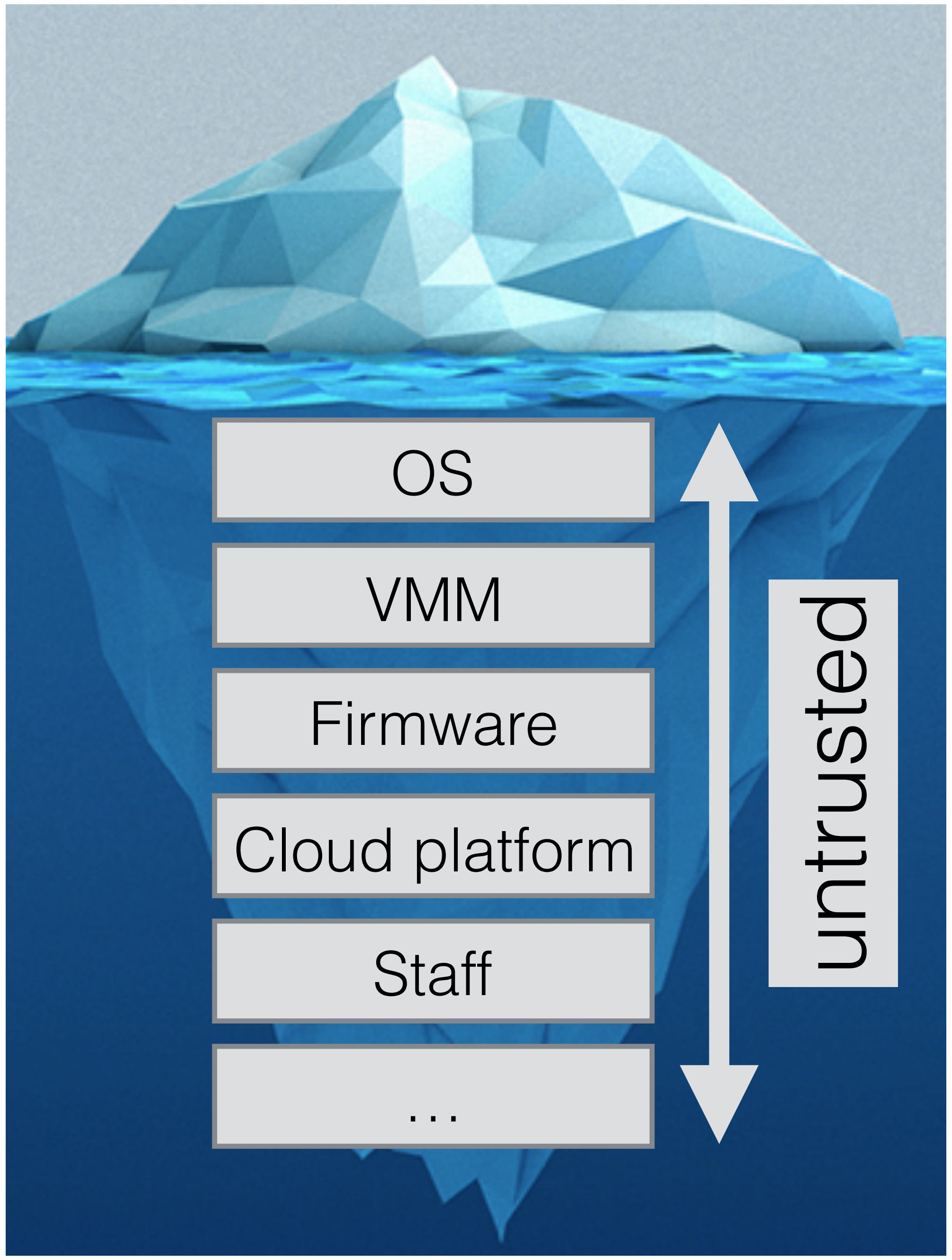
- run unmodified Linux applications …

We want to …



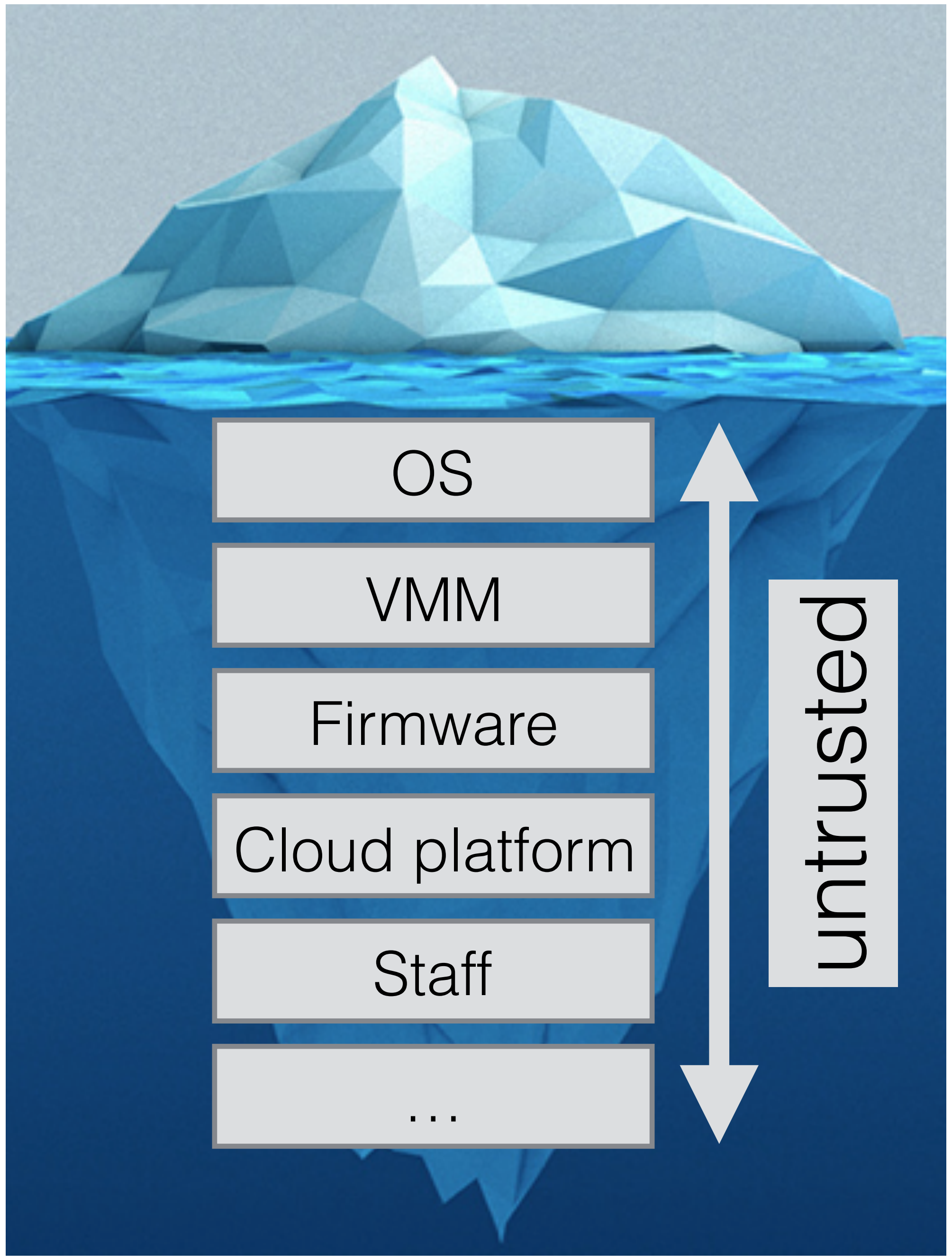
- run unmodified Linux applications …
- in containers …

We want to …



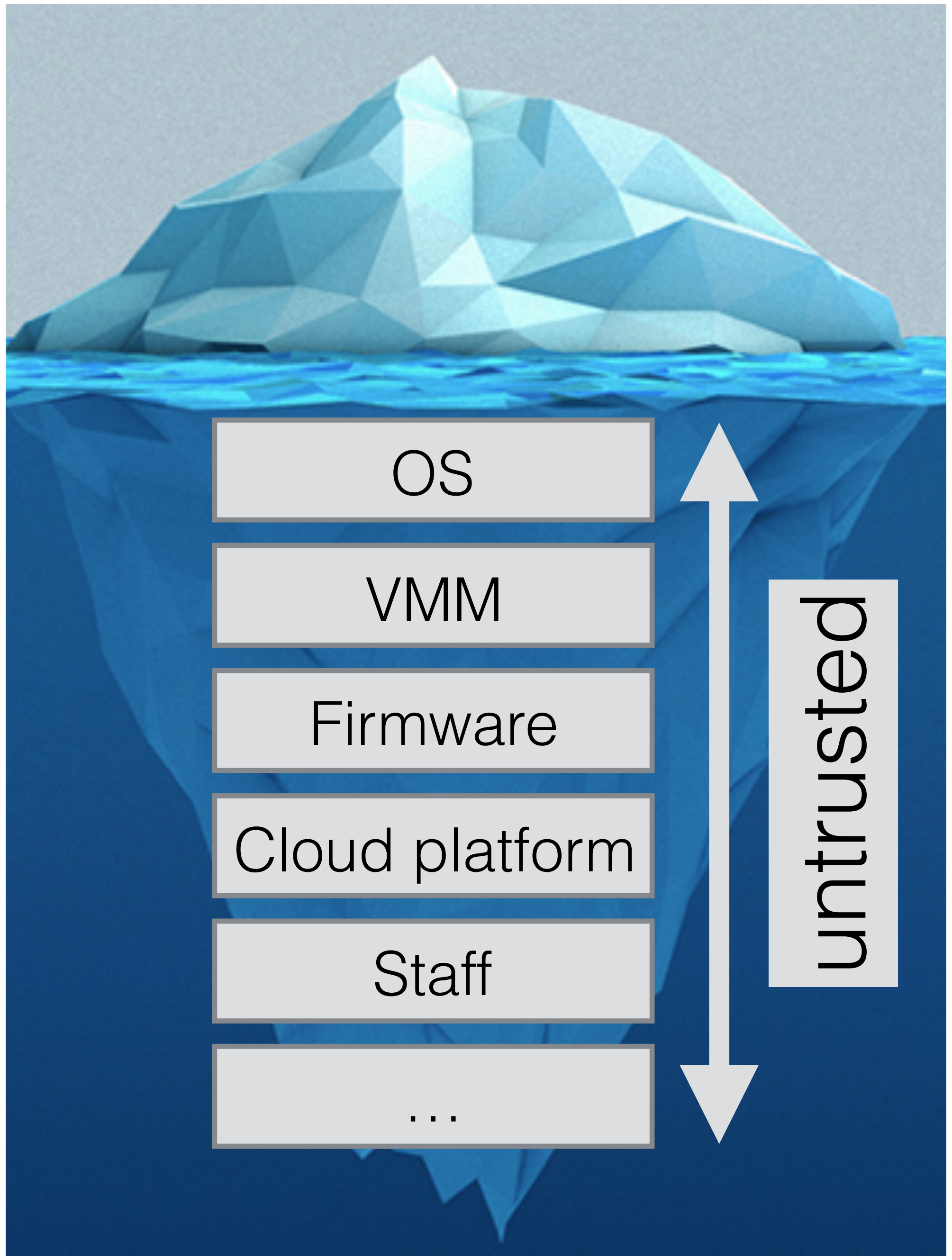
- run unmodified Linux applications …
- in containers …
- in an untrusted cloud …

We want to …



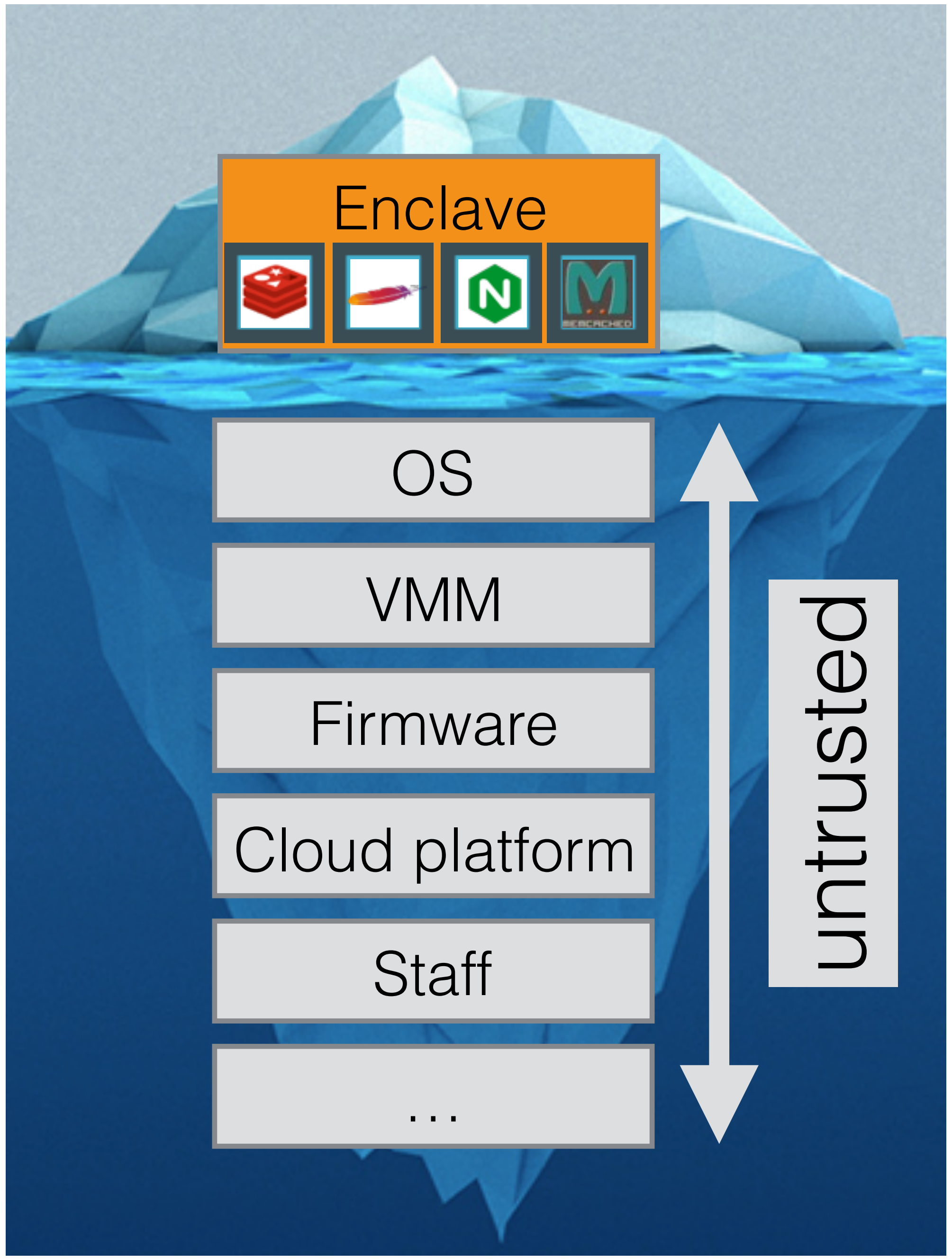
- run unmodified Linux applications …
- in containers …
- in an untrusted cloud …
- securely and …

We want to …



- run unmodified Linux applications …
- in containers …
- in an untrusted cloud …
- securely and …
- with acceptable performance

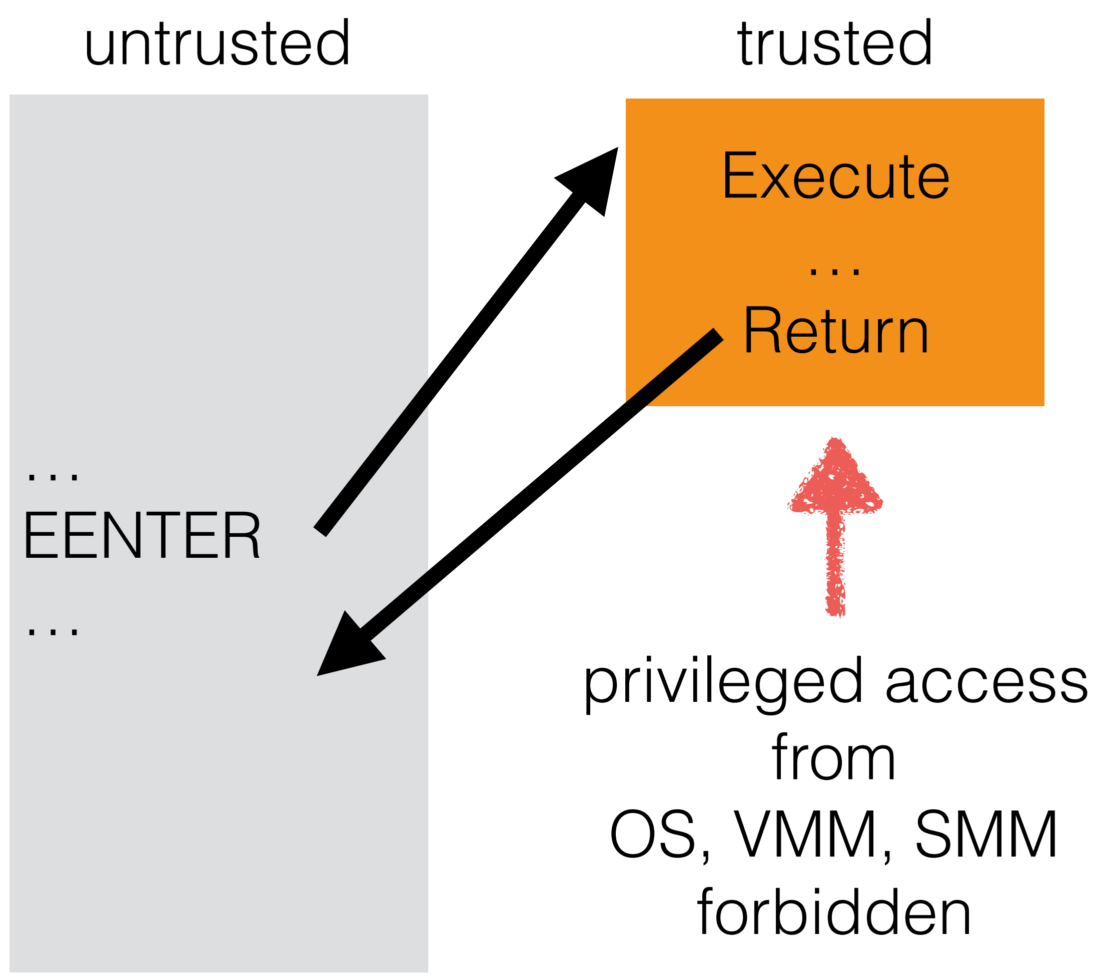
Secure Guard Extensions



- New **enclave** processor **mode**
- Users can create a HW-enforced trusted environment
- Only trust Intel and Secure Guard Extensions (SGX) implementation

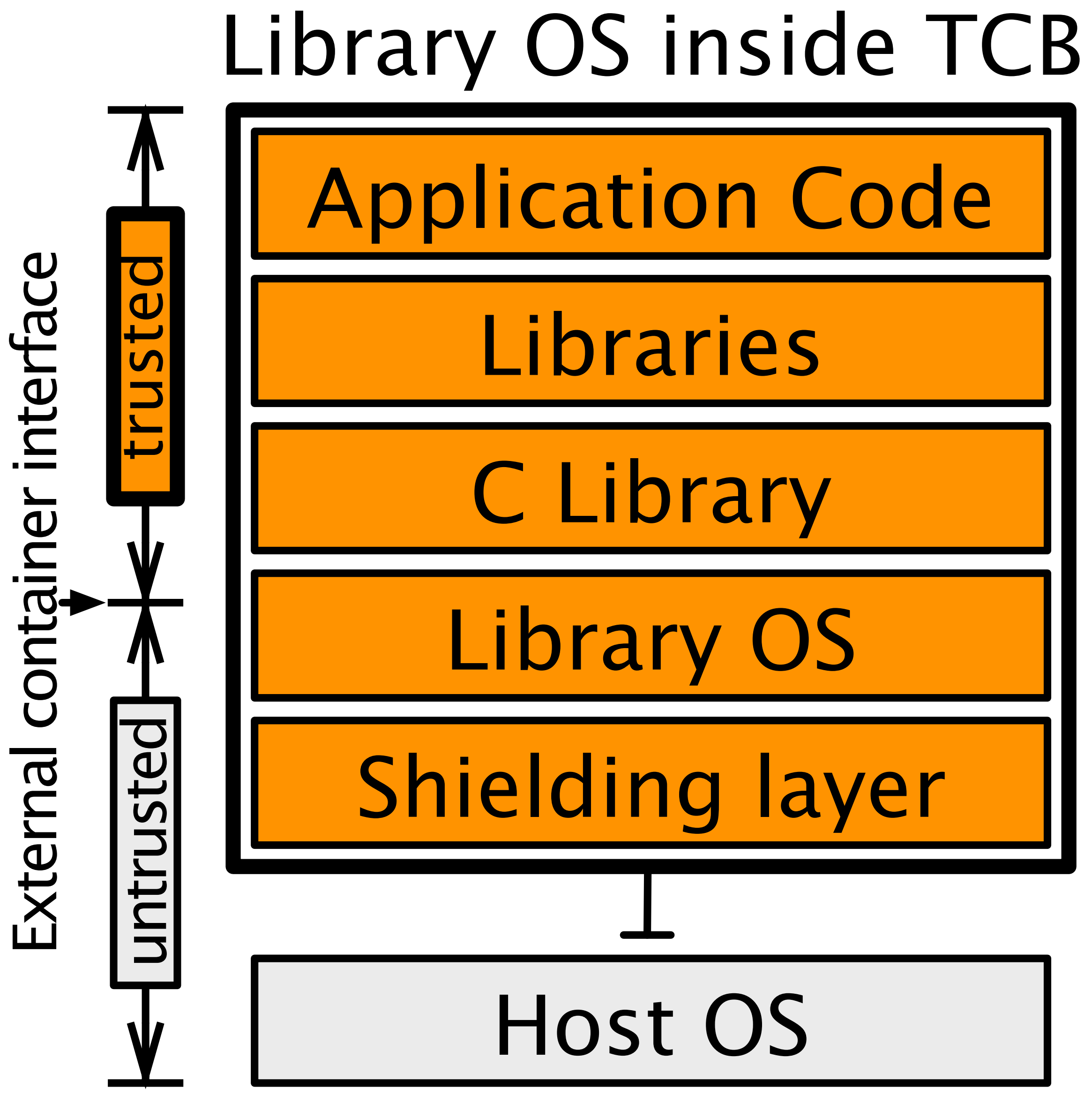
SGX: HW-enforced Security

- 18 new instructions to manage enclave life cycle
- **Enclave memory** only accessible from enclave
- Certain instructions disallowed, e.g., `syscall`



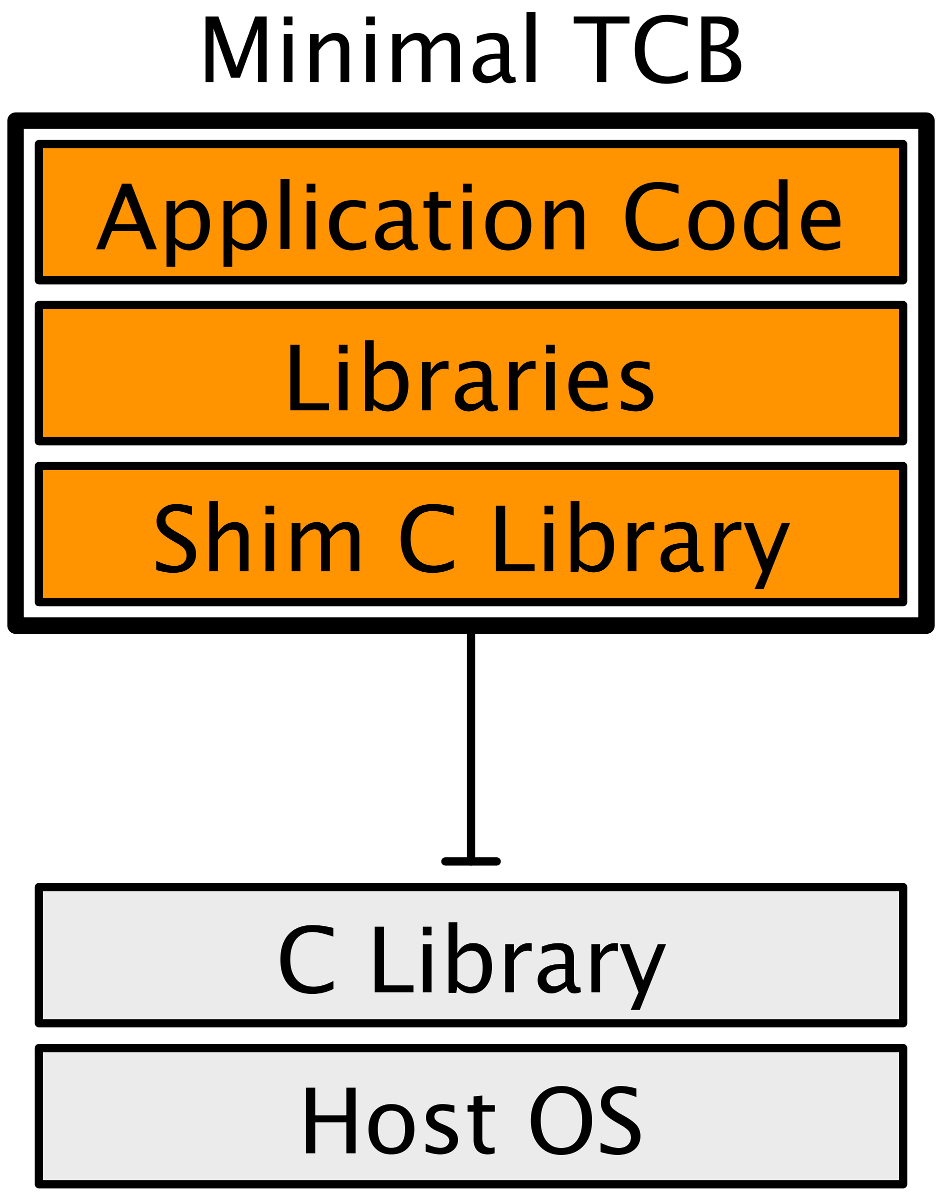
Challenge 1: Interface

- Haven (OSDI'14): library operating system in enclave
- Large TCB → more vulnerable
- Small interface (22 system calls)
- Shields protect the interface

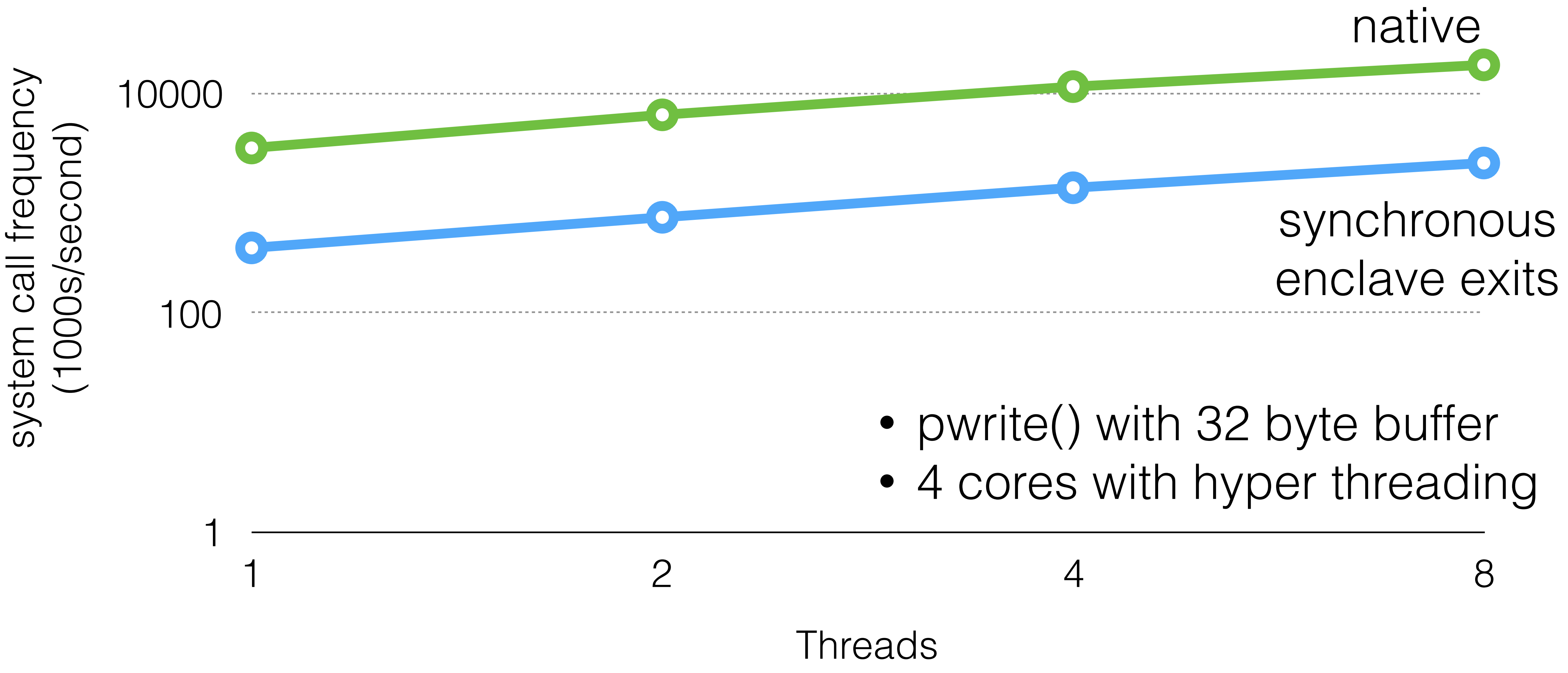


Challenge 1: Interface

- Small TCB
- C library interface is complex
- Harder to protect

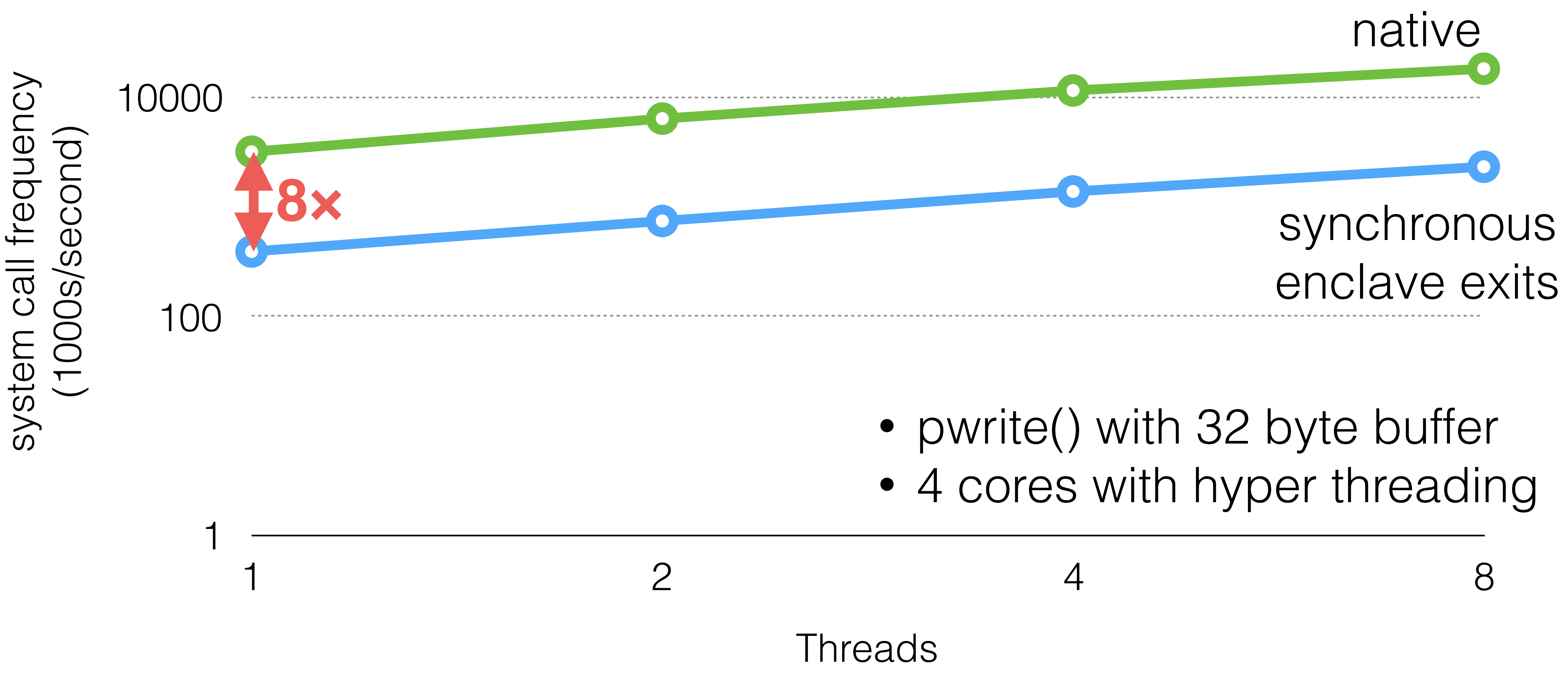


Challenge 2: Performance



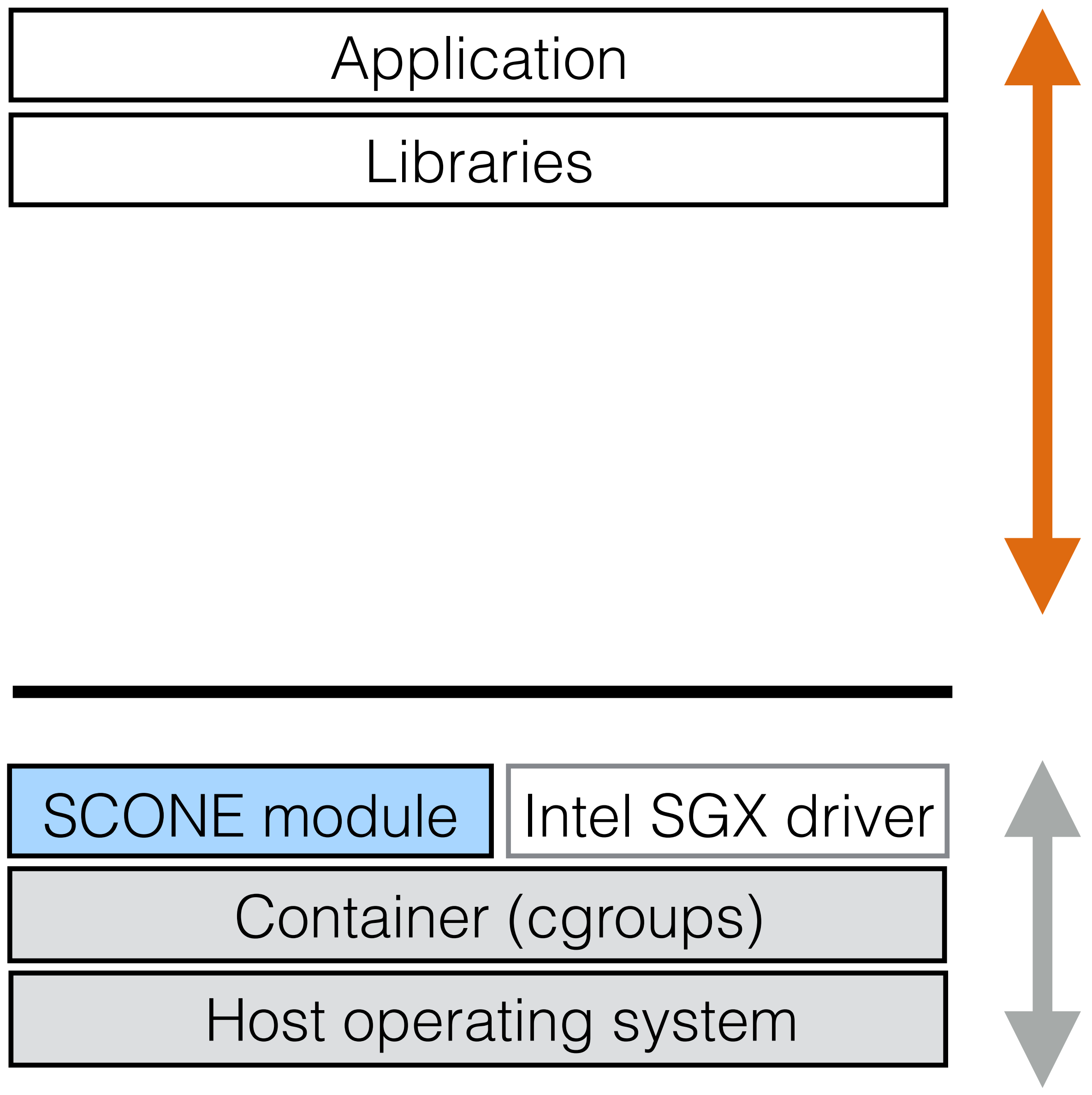
- pwrite() with 32 byte buffer
- 4 cores with hyper threading

Challenge 2: Performance



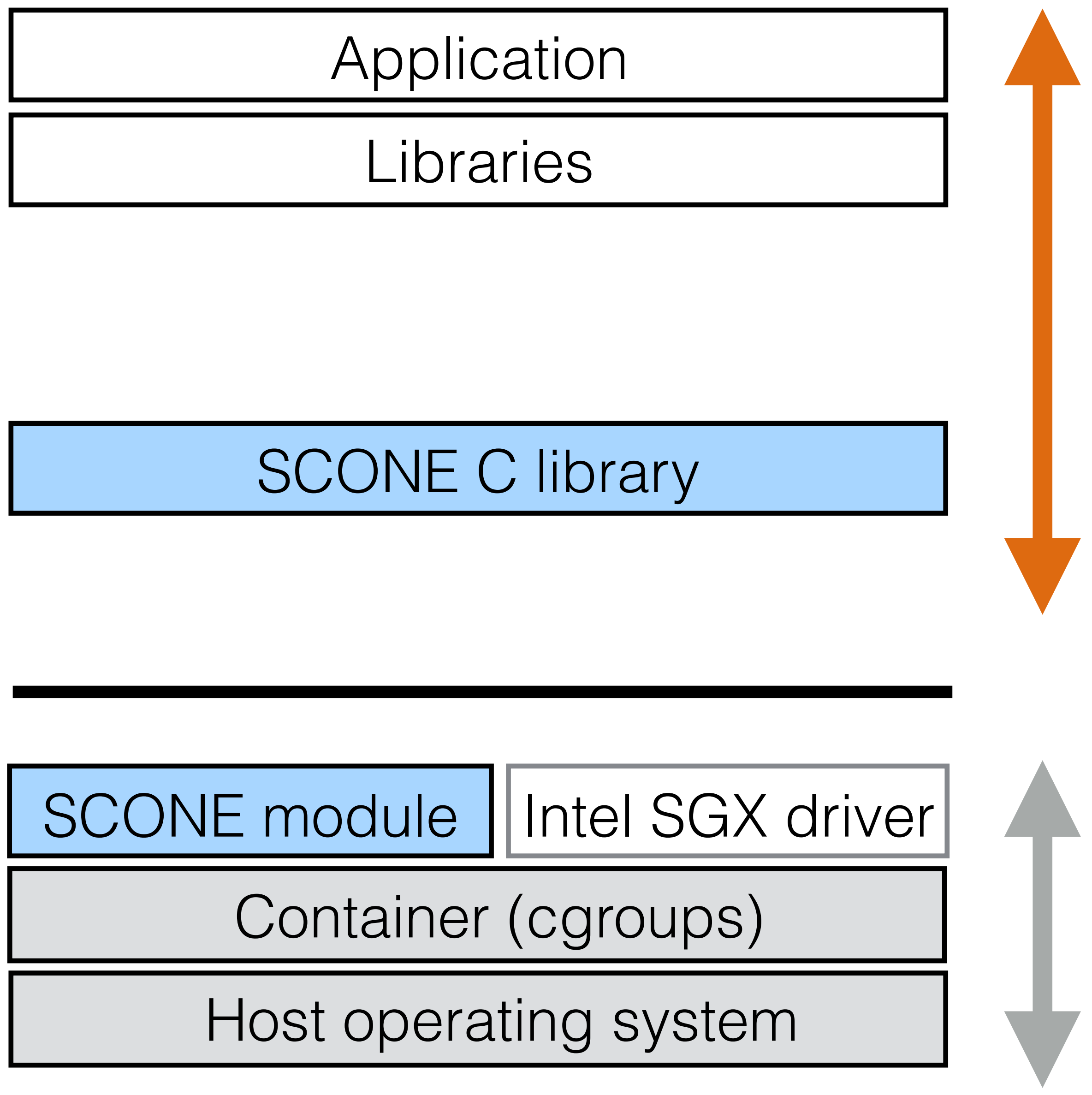
- pwrite() with 32 byte buffer
- 4 cores with hyper threading

SCONE Architecture



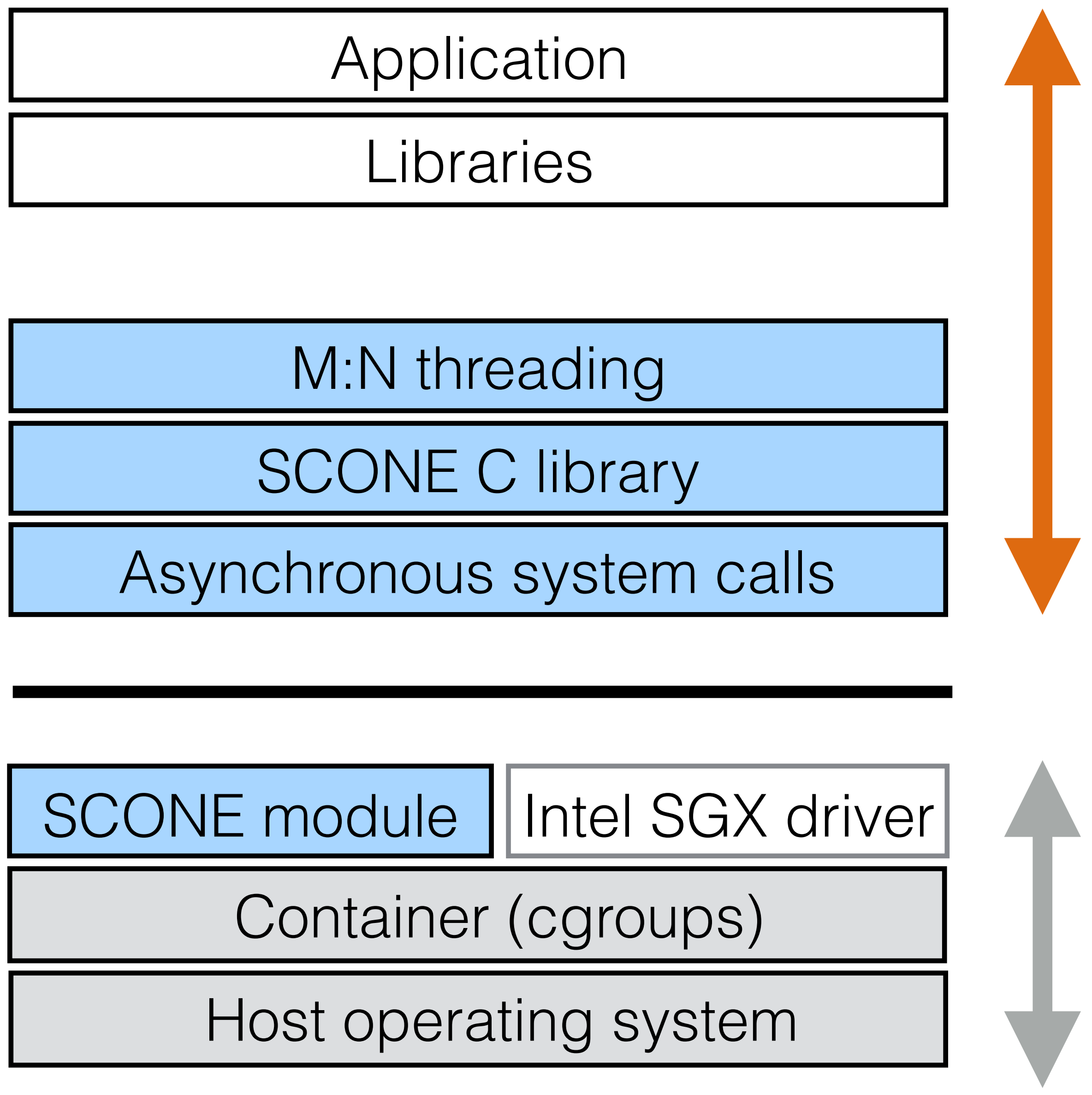
SCONE Architecture

- Enhanced C library → small TCB (Challenge 1)



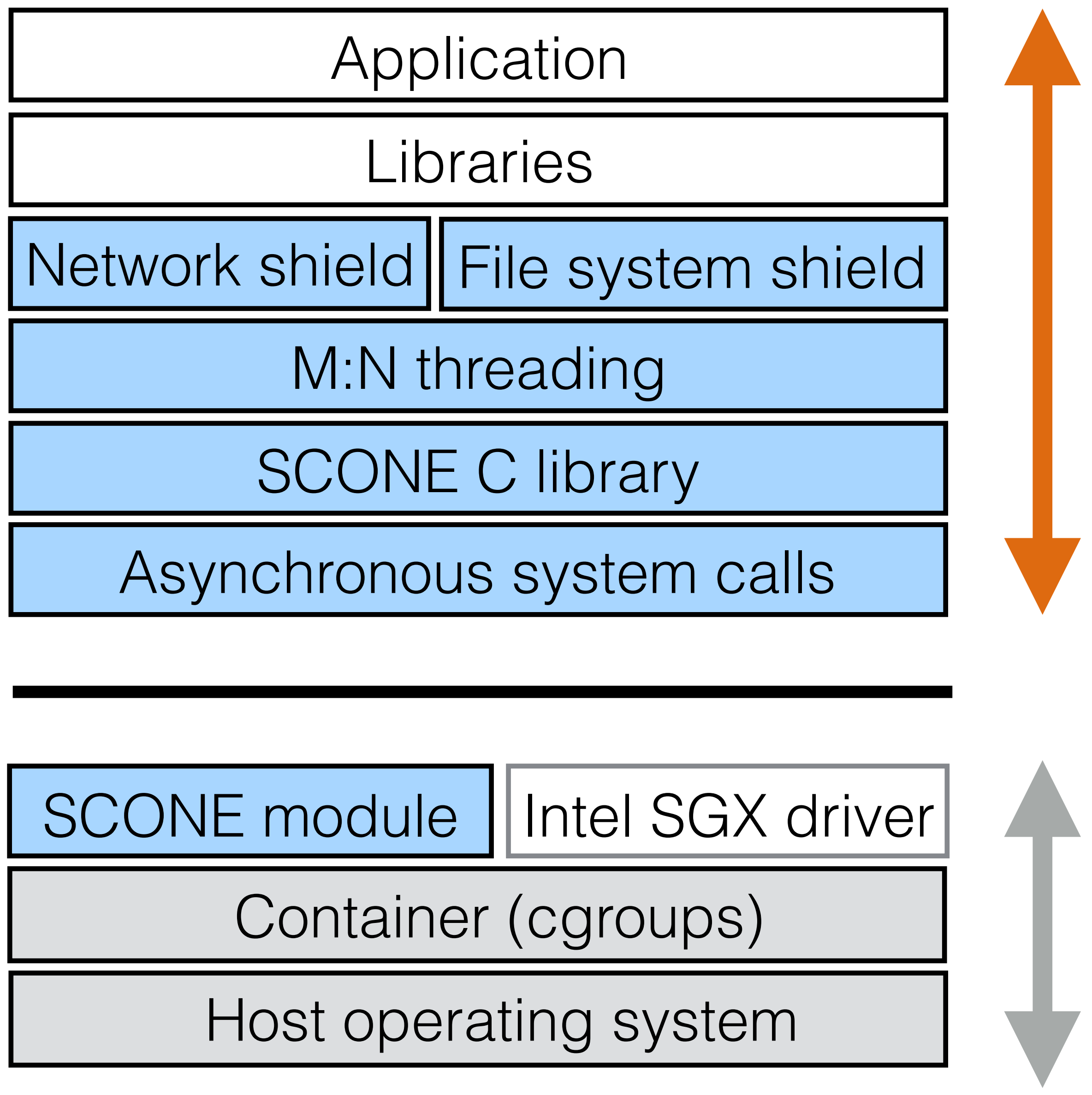
SCONE Architecture

- Enhanced C library → small TCB (Challenge 1)
- Asynchronous system calls and user space threading **reduce** number of **enclave exits** (Challenge 2)



SCONE Architecture

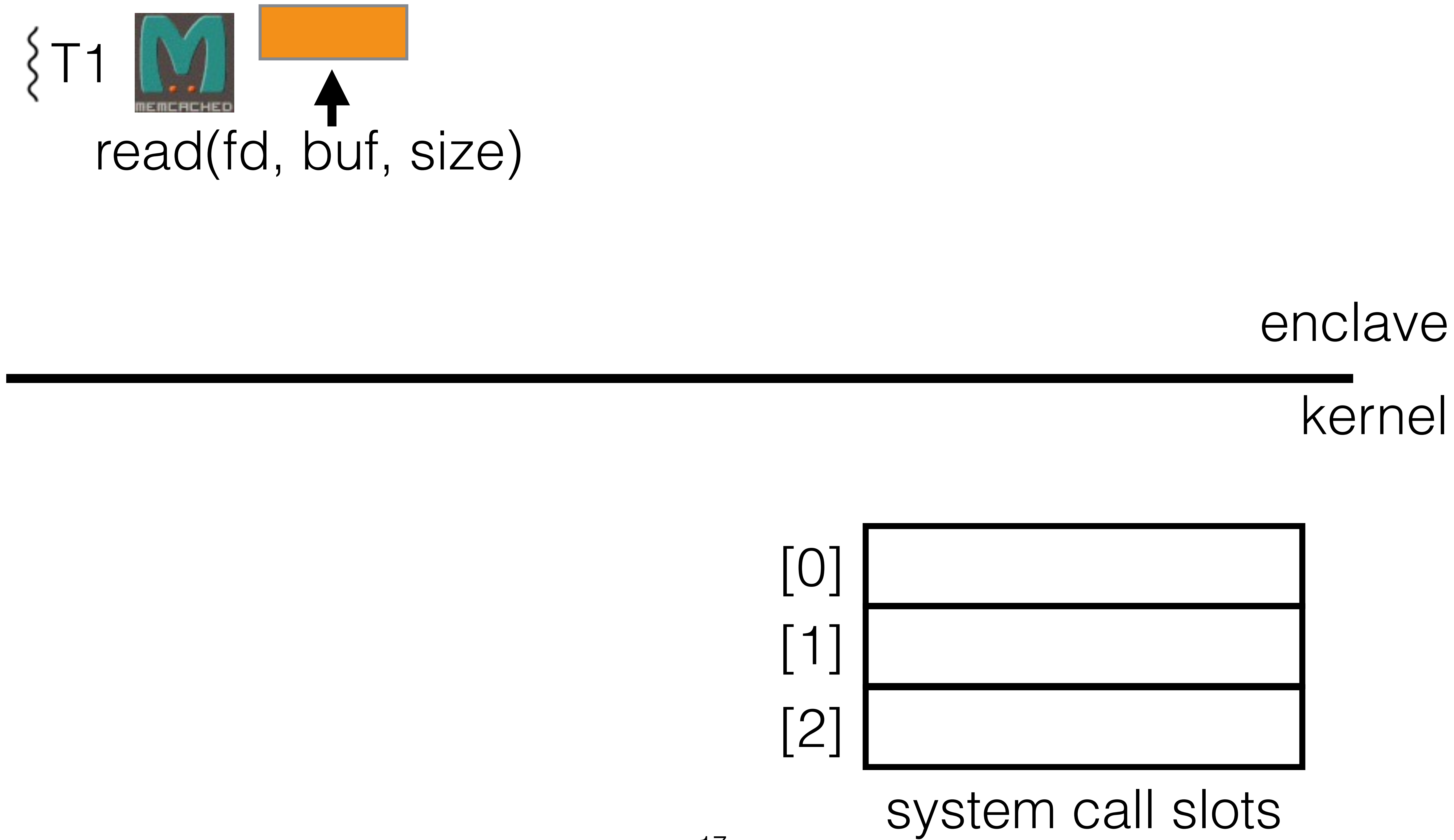
- Enhanced C library → small TCB (Challenge 1)
- Asynchronous system calls and user space threading **reduce** number of **enclave exits** (Challenge 2)
- Network and file system shields **actively** protect user data



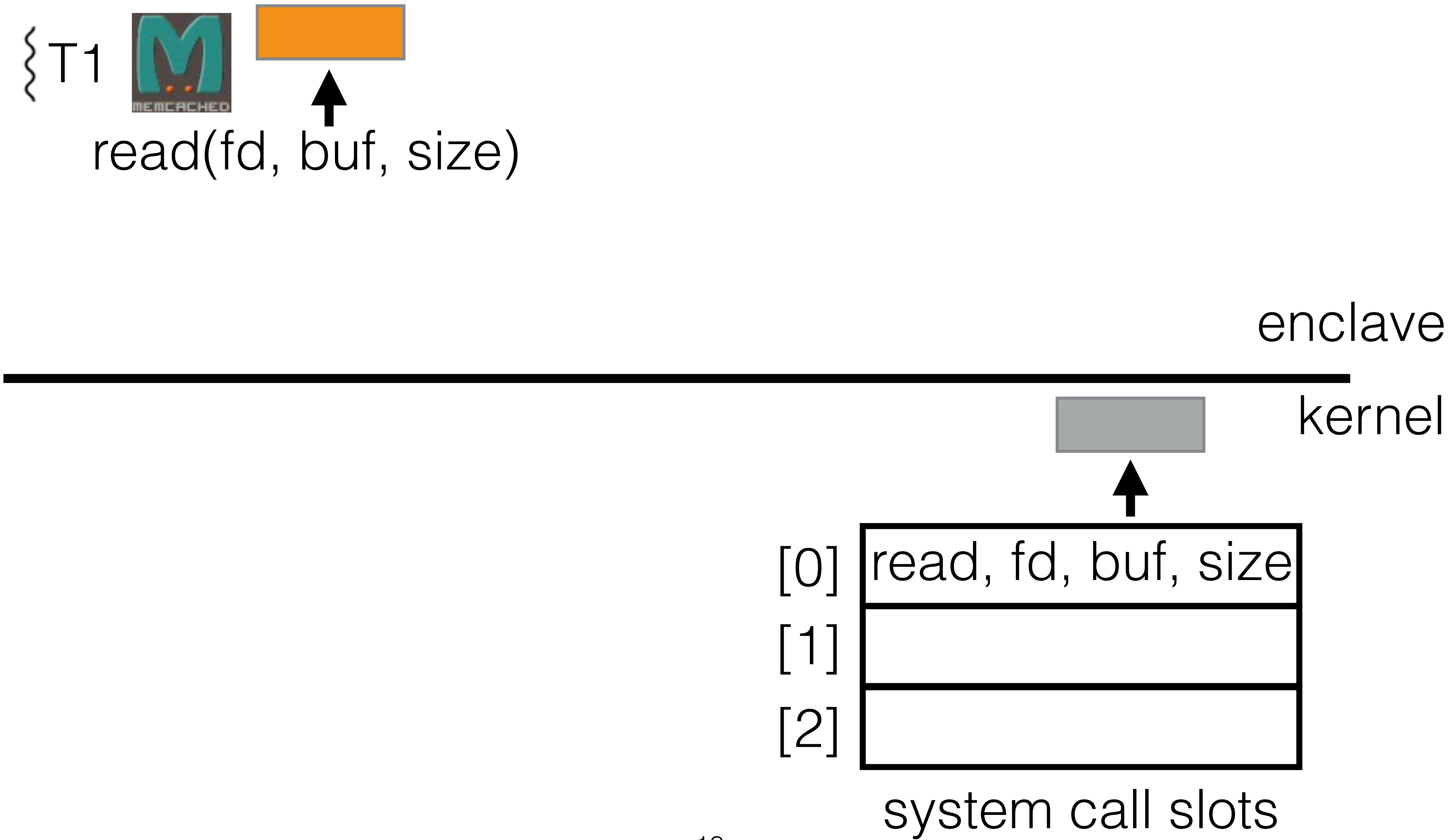
Anatomy of a System Call

enclave
kernel

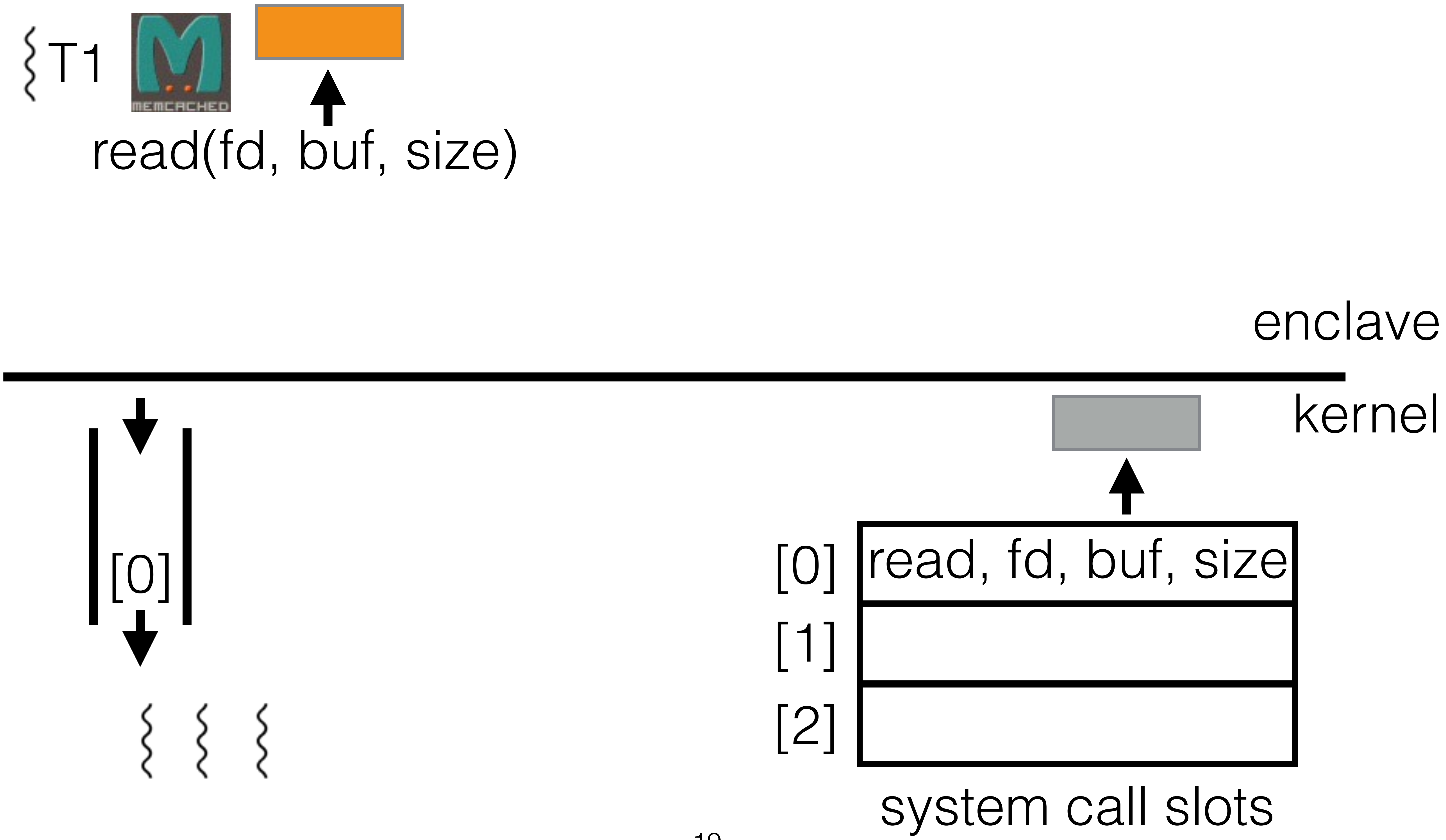
Anatomy of a System Call



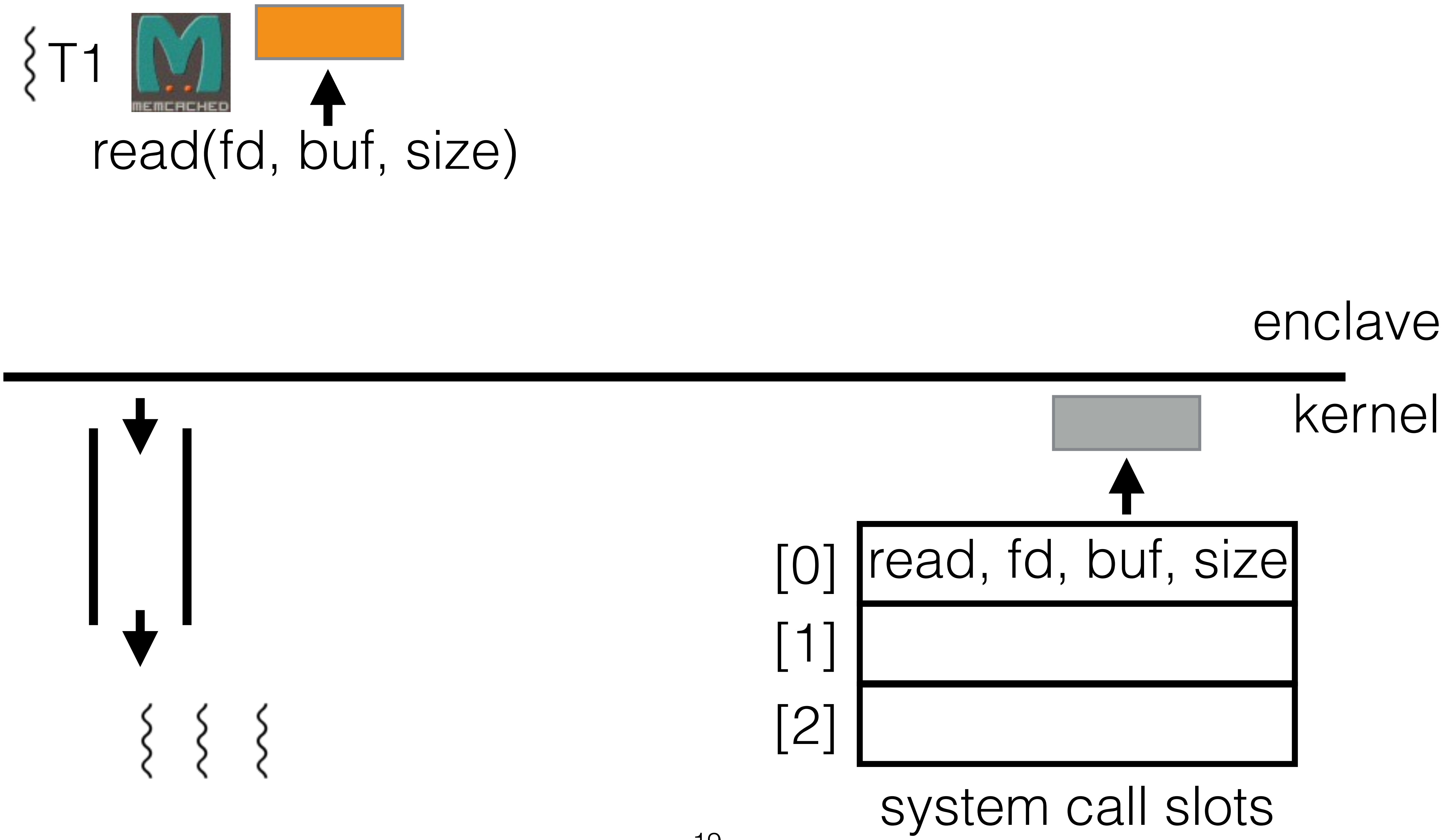
Anatomy of a System Call



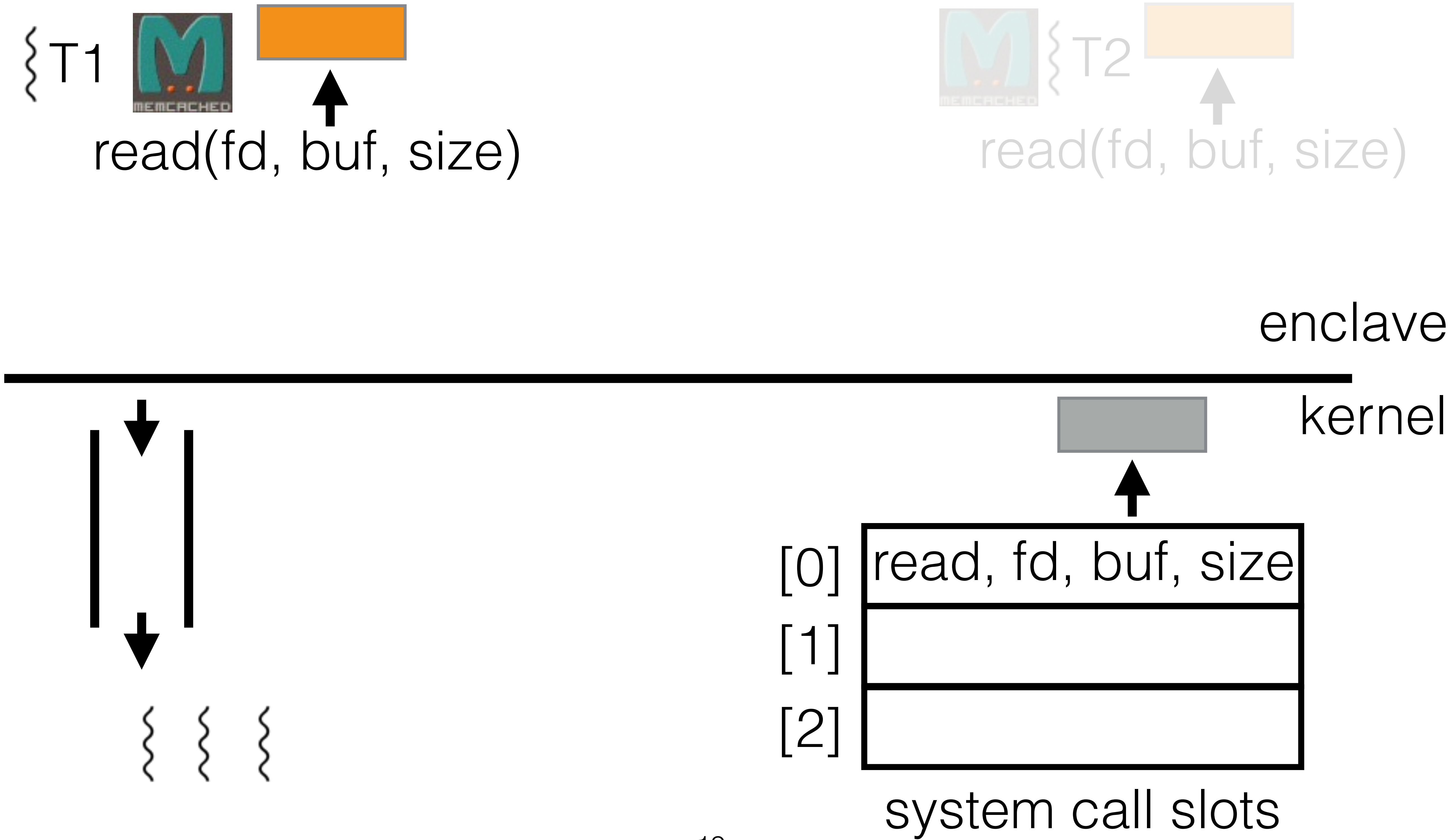
Anatomy of a System Call



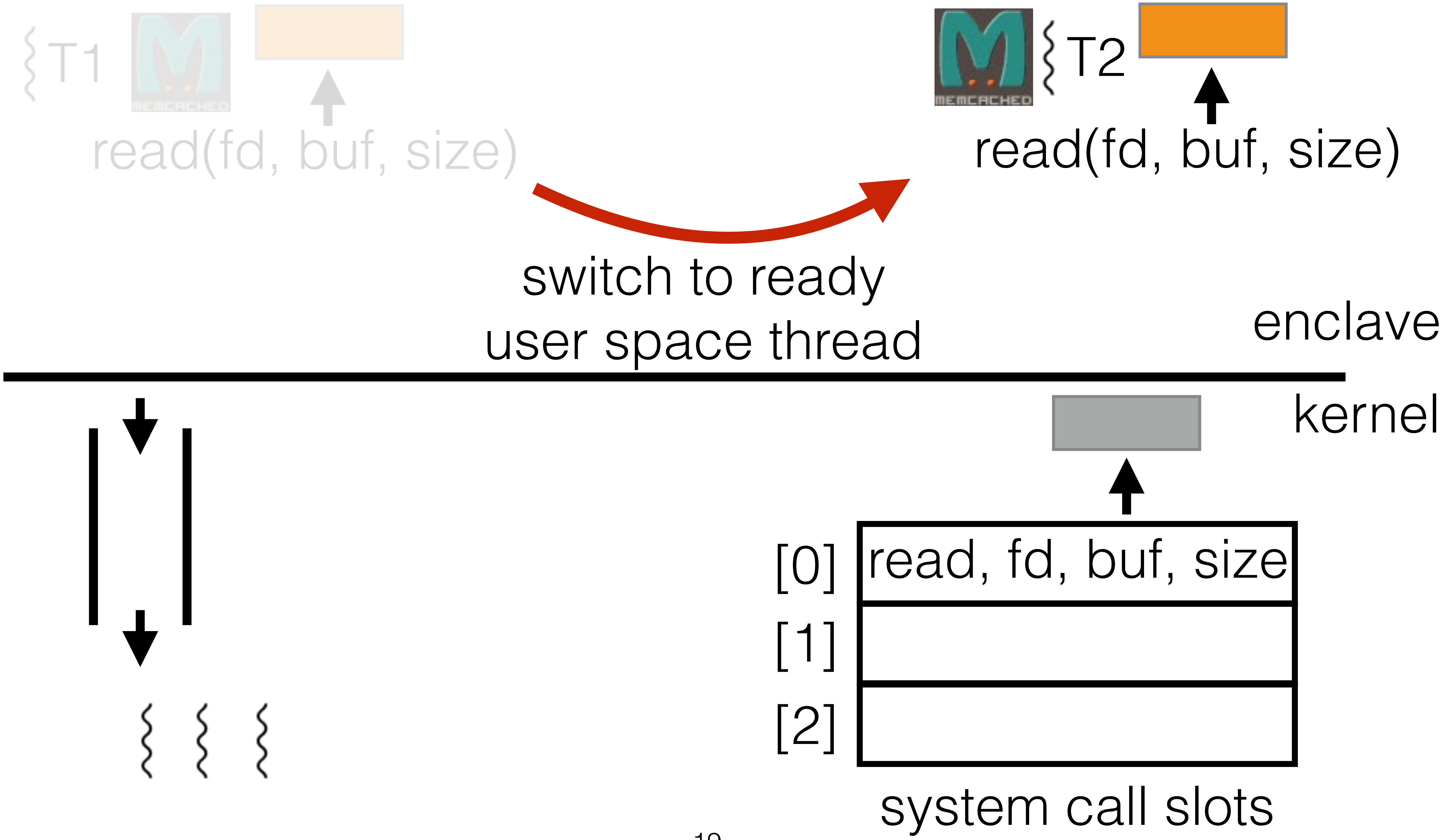
Anatomy of a System Call



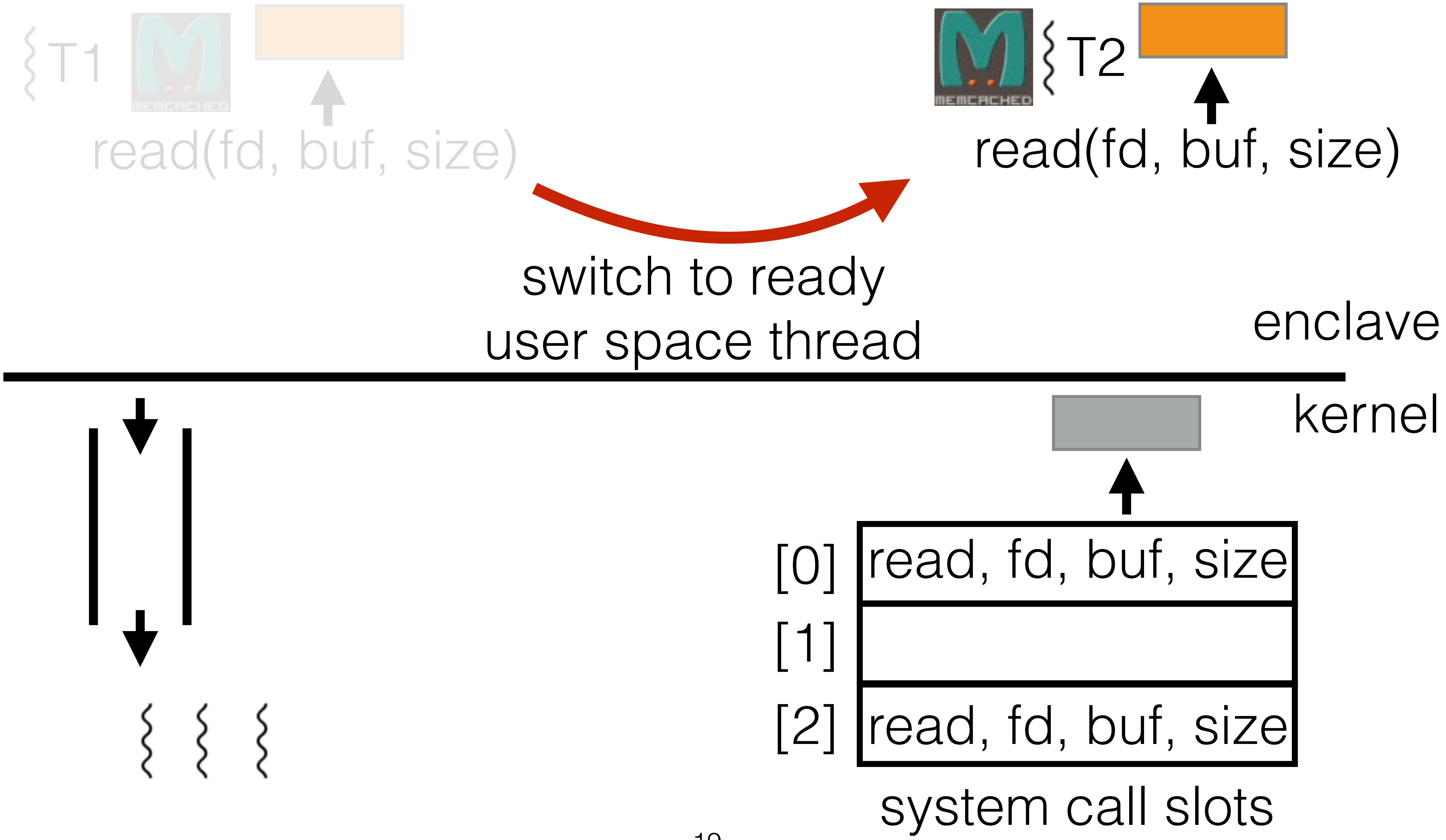
Anatomy of a System Call



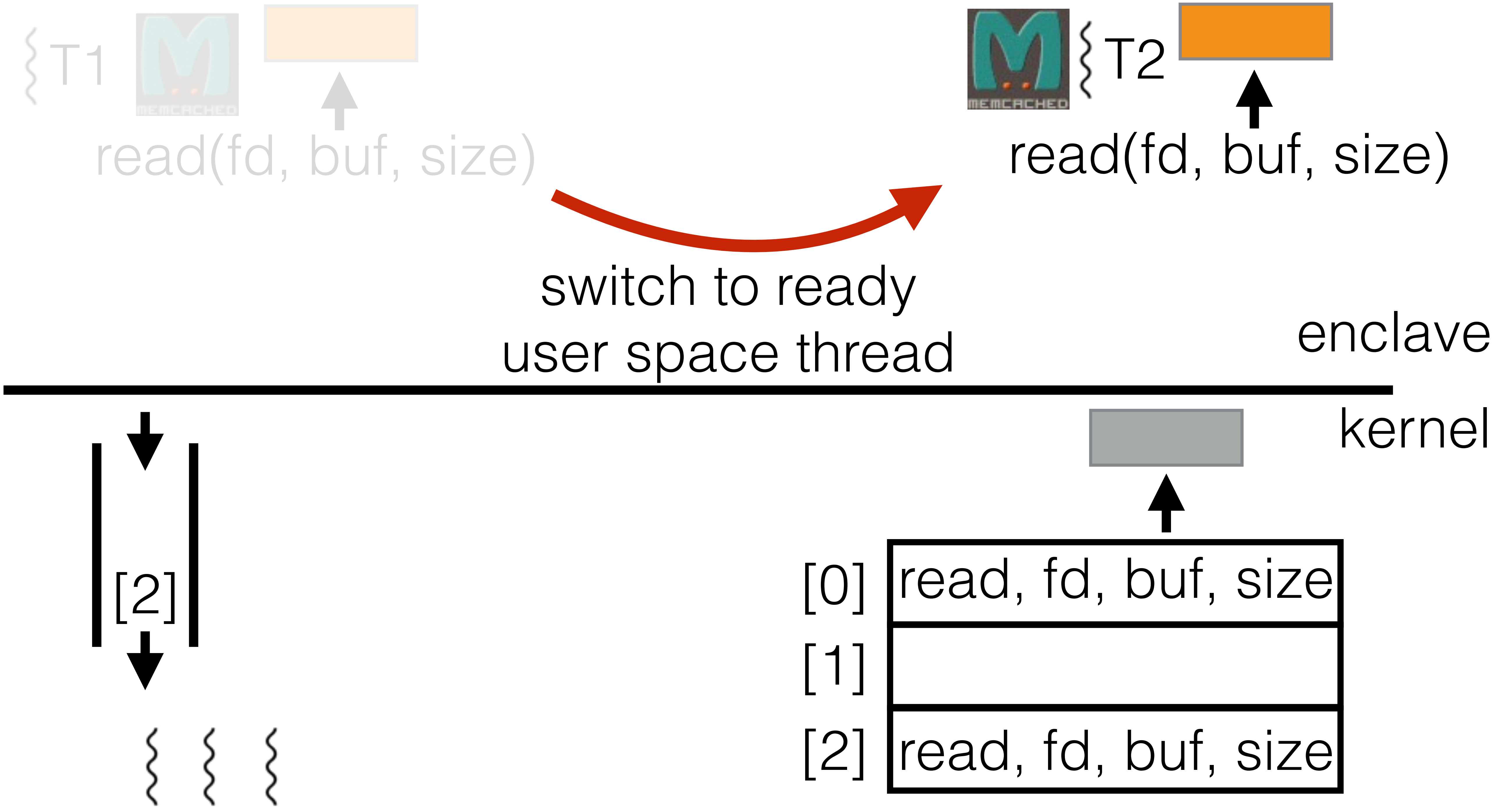
Anatomy of a System Call



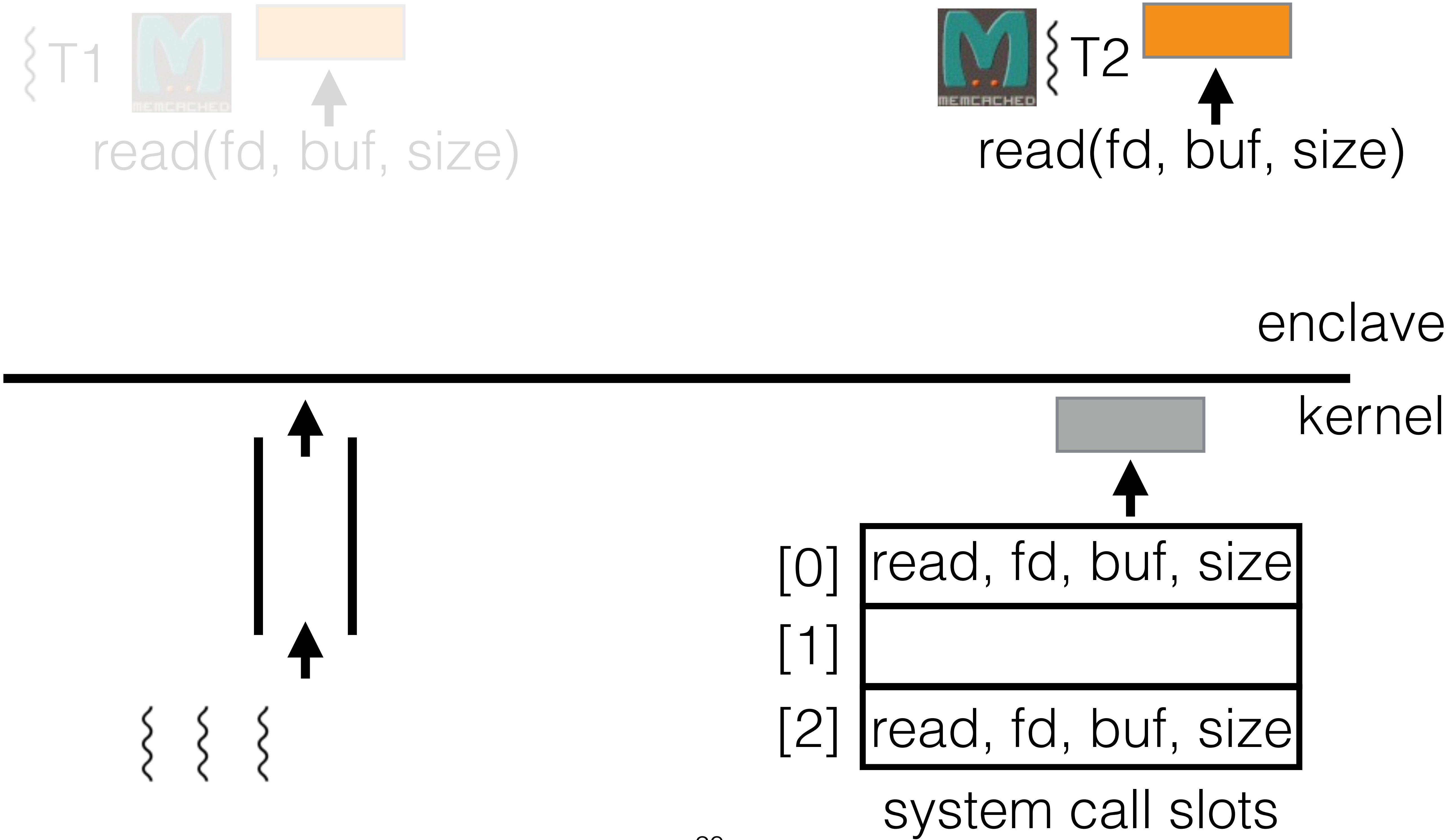
Anatomy of a System Call



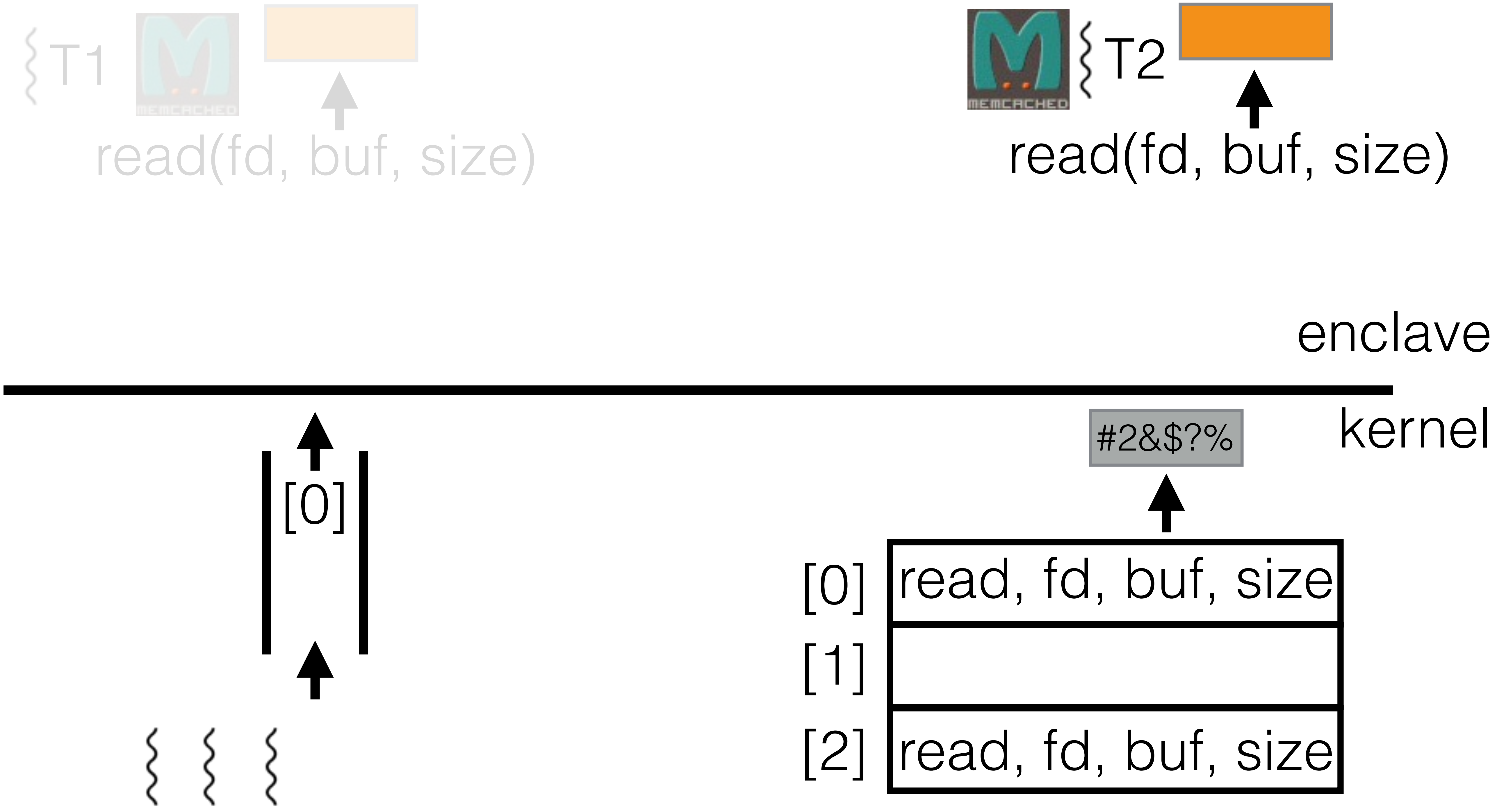
Anatomy of a System Call



Anatomy of a System Call

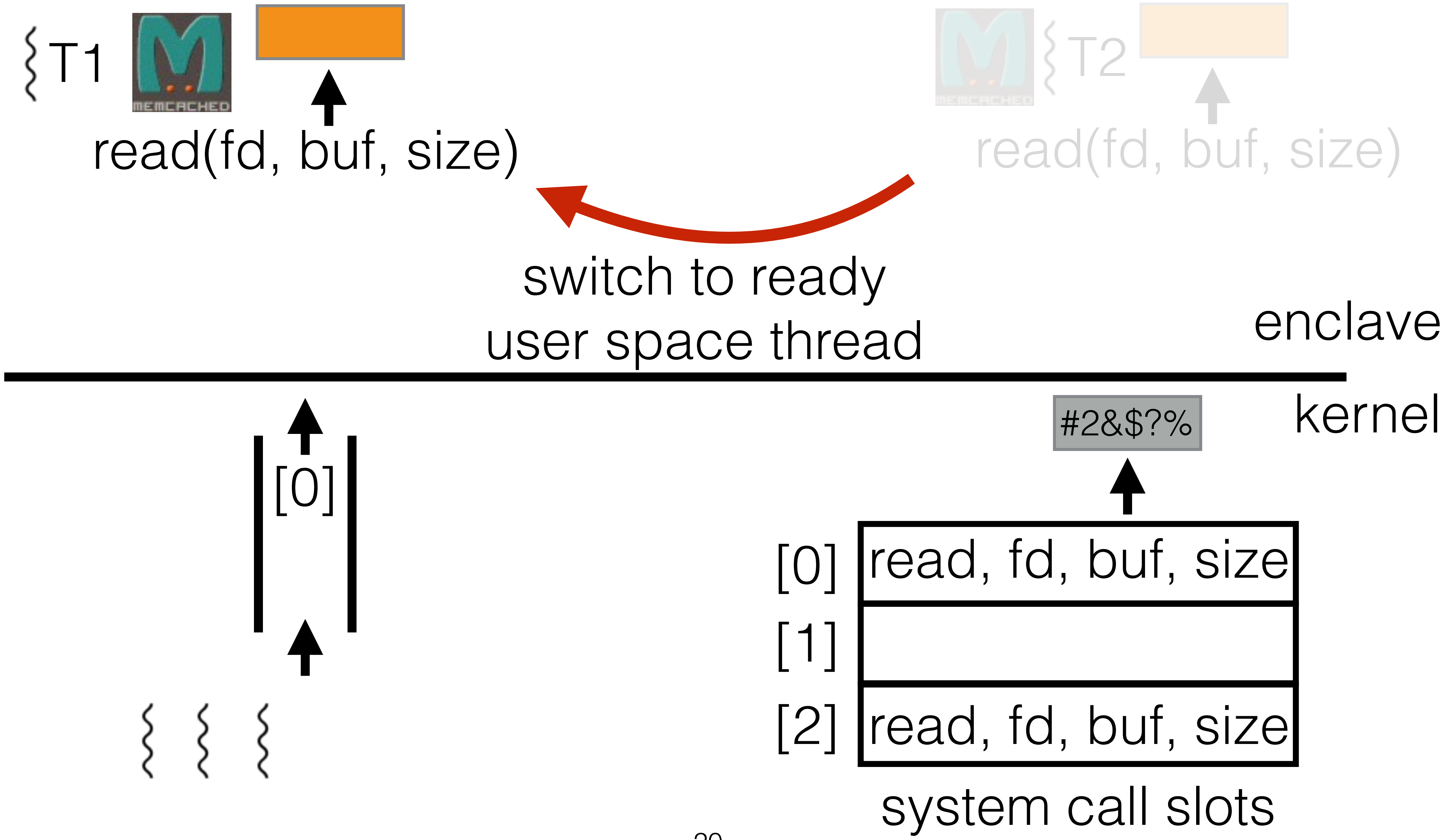


Anatomy of a System Call

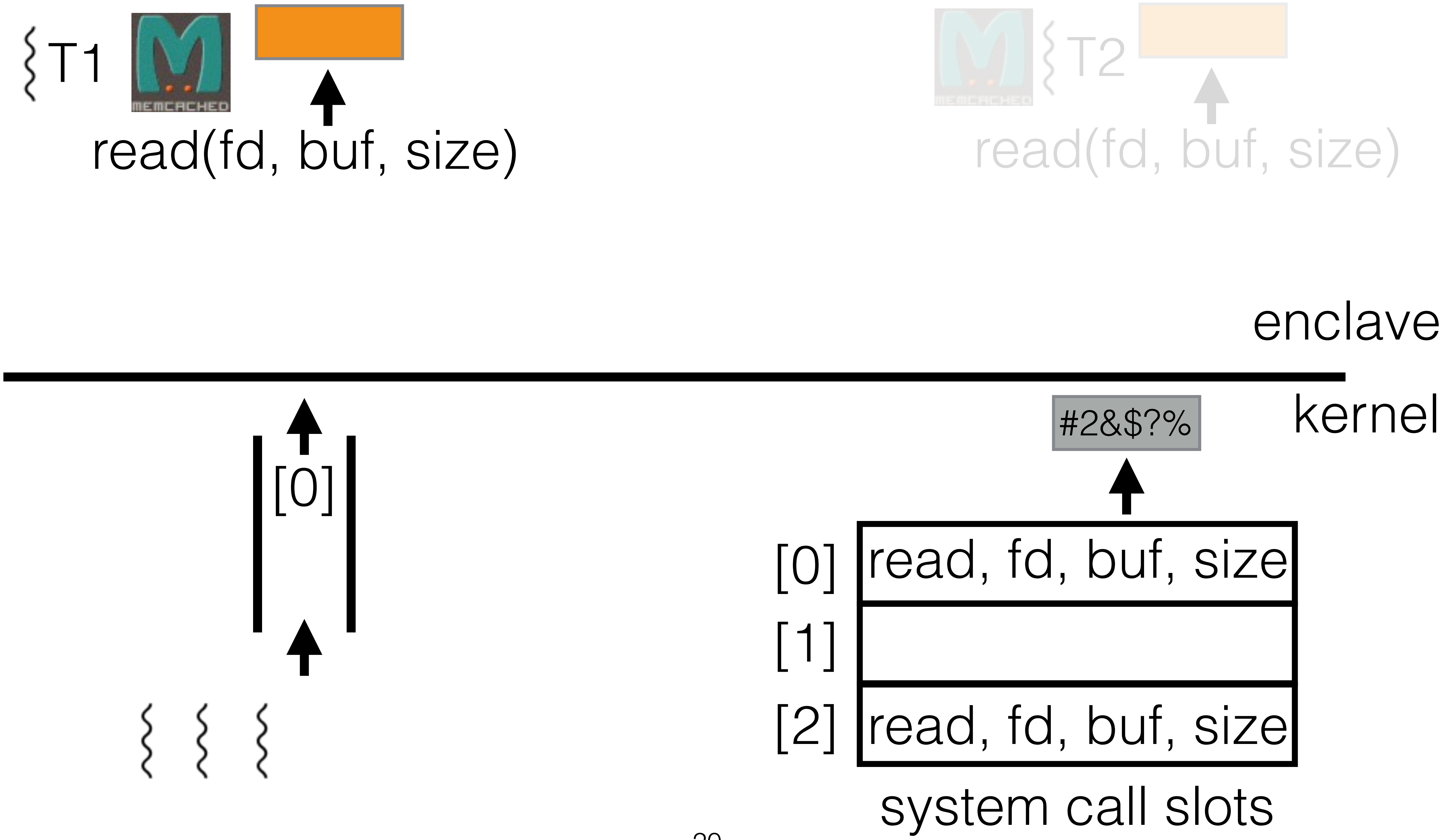


system call slots

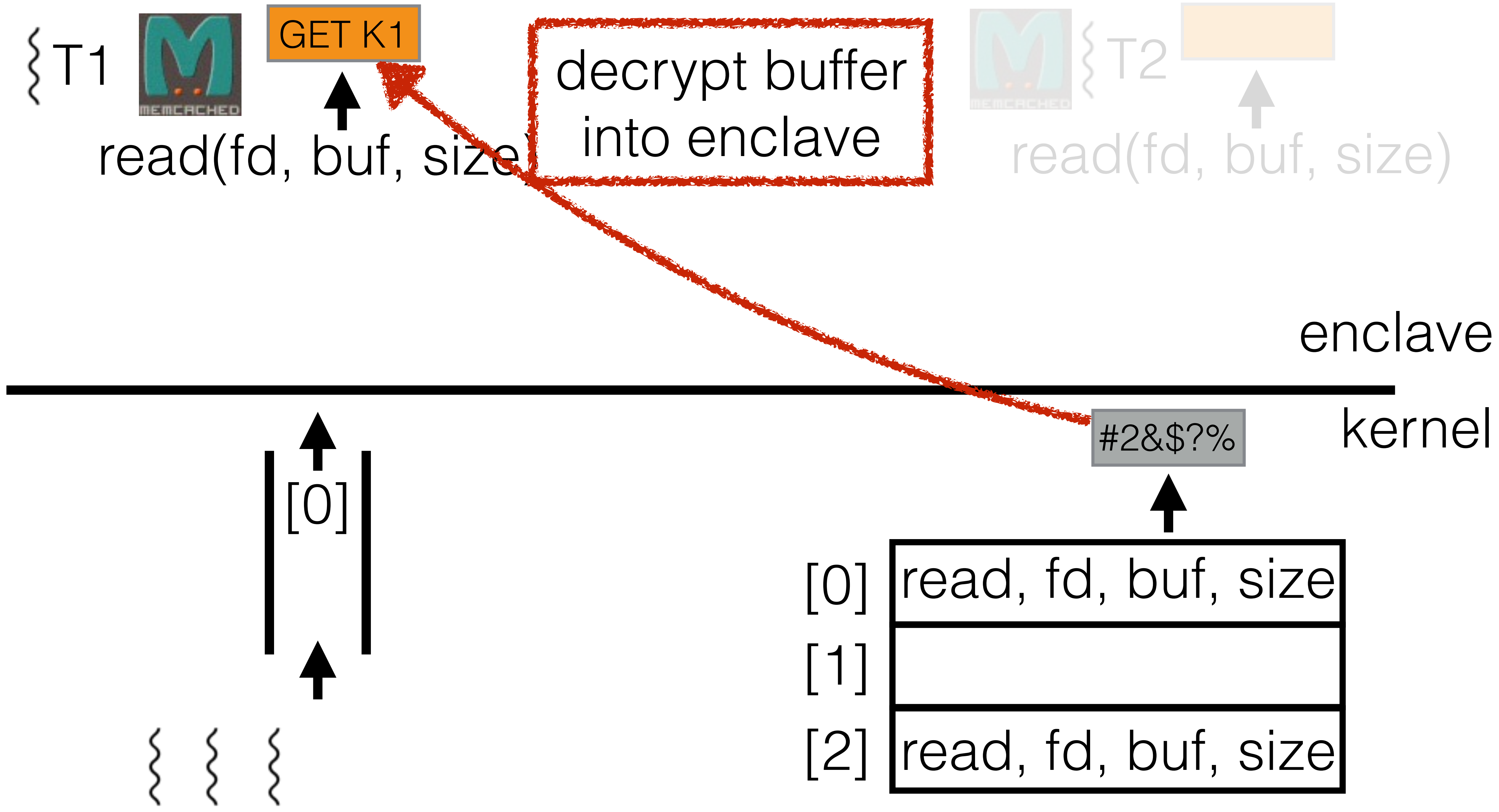
Anatomy of a System Call



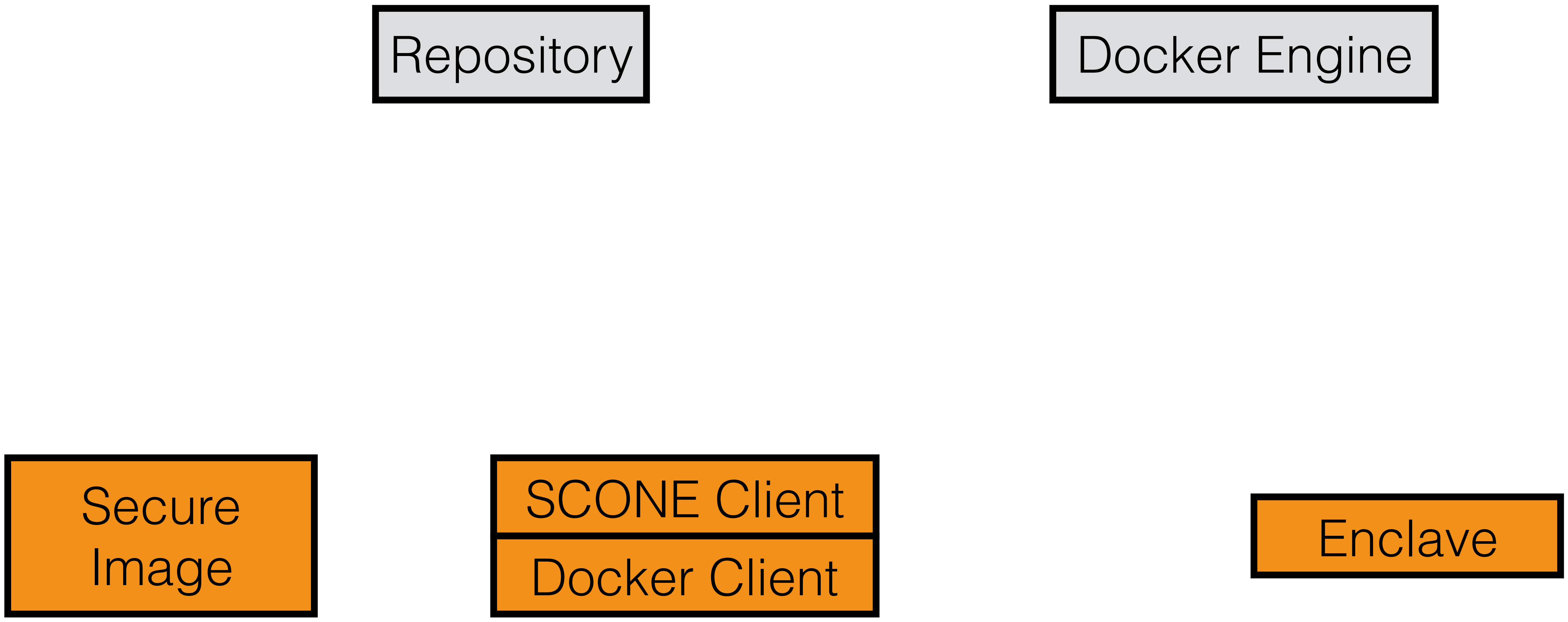
Anatomy of a System Call



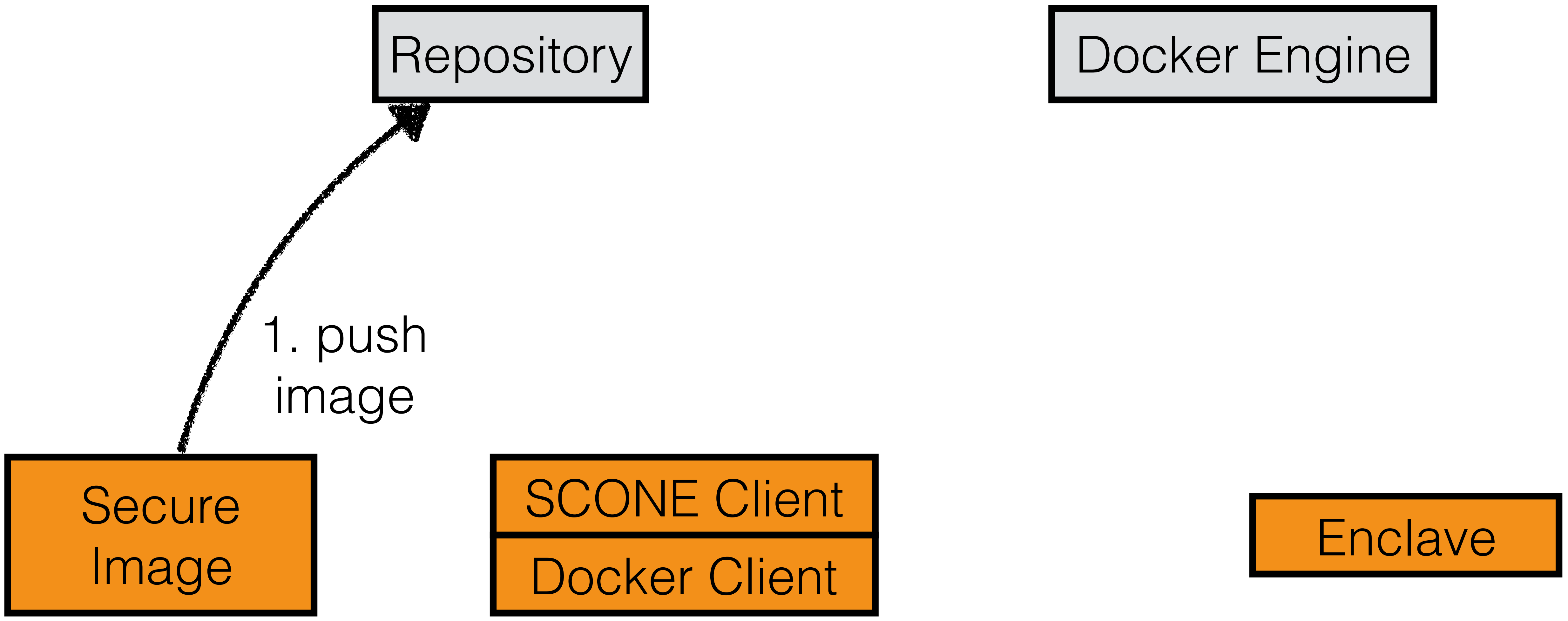
Anatomy of a System Call



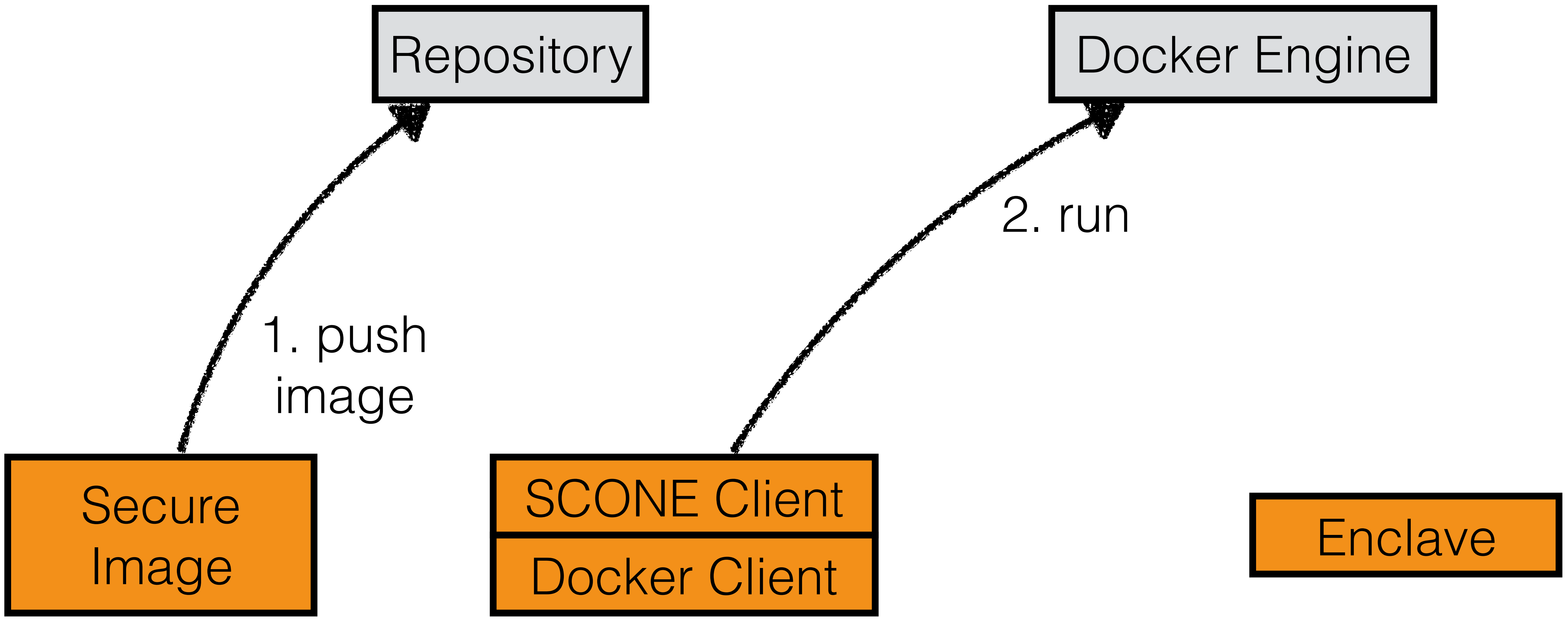
Container Integration



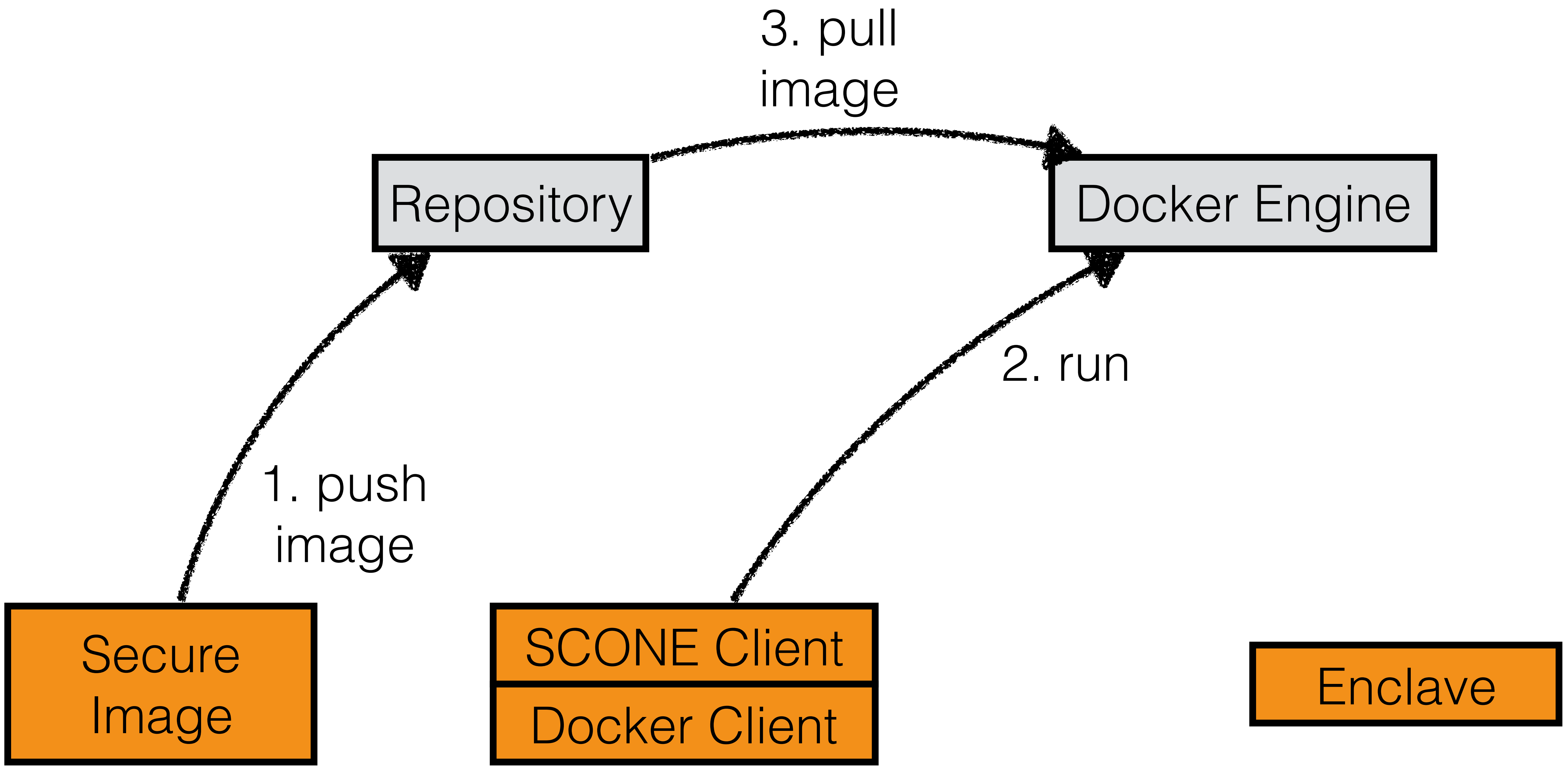
Container Integration



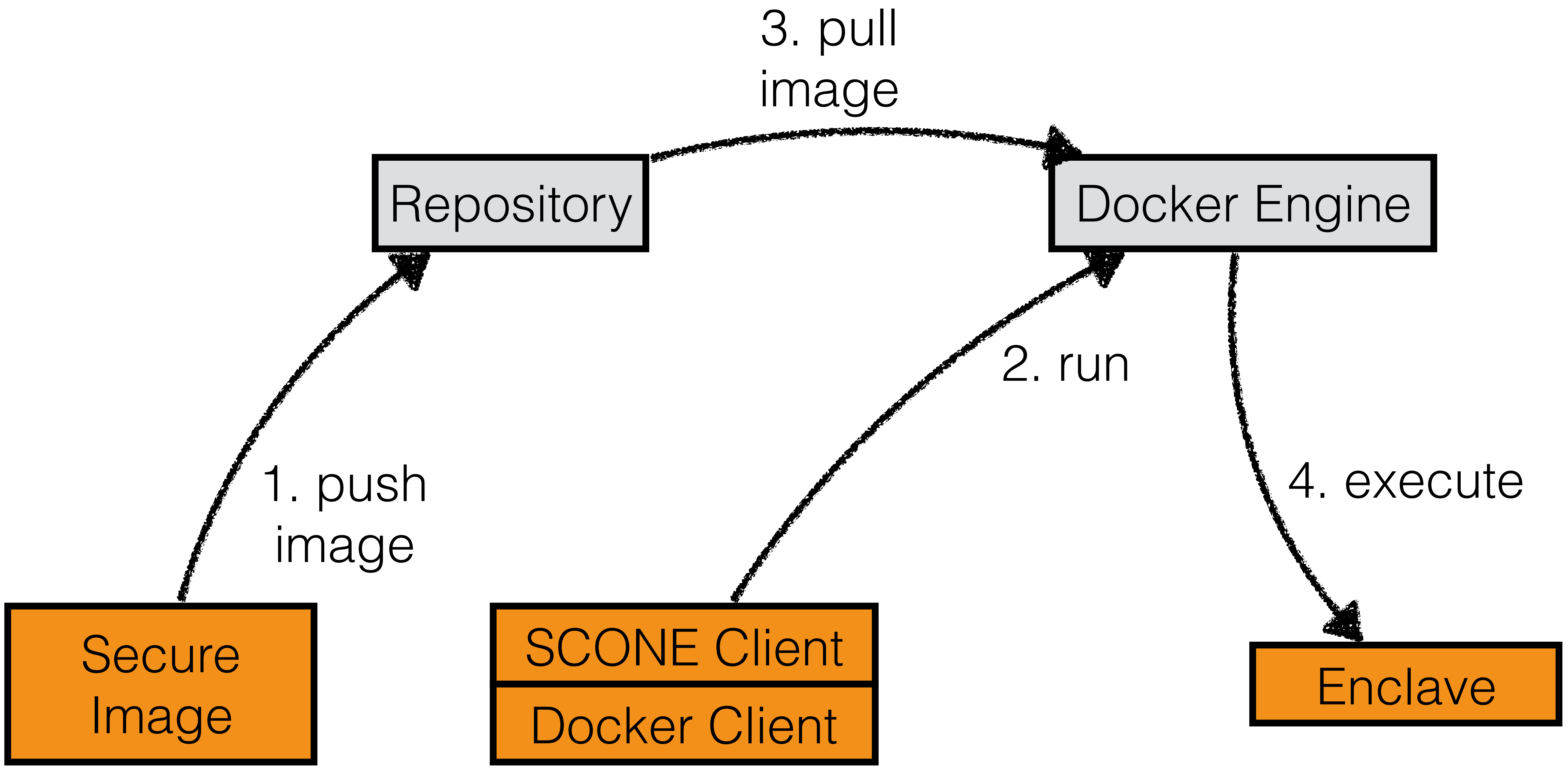
Container Integration



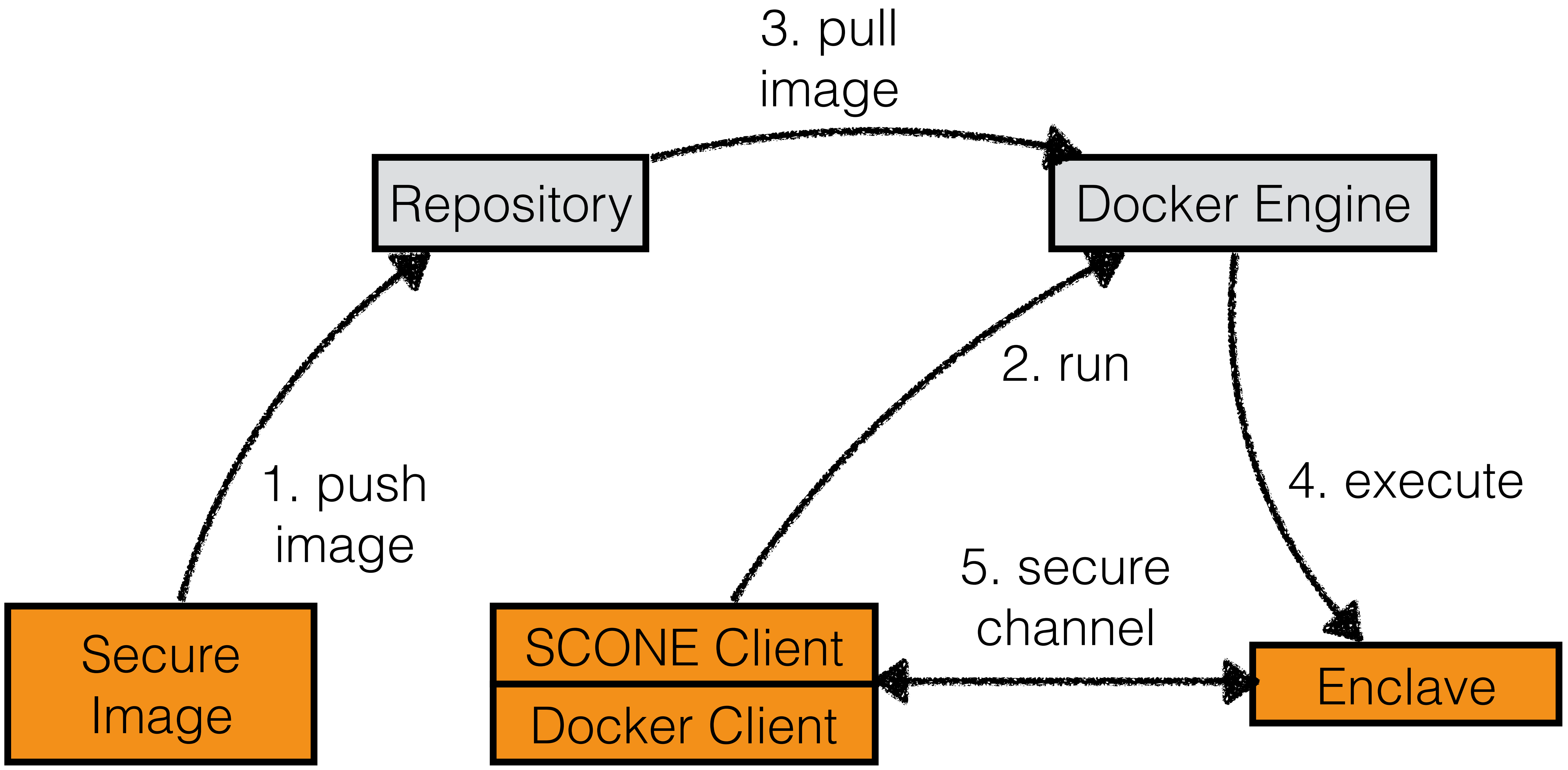
Container Integration



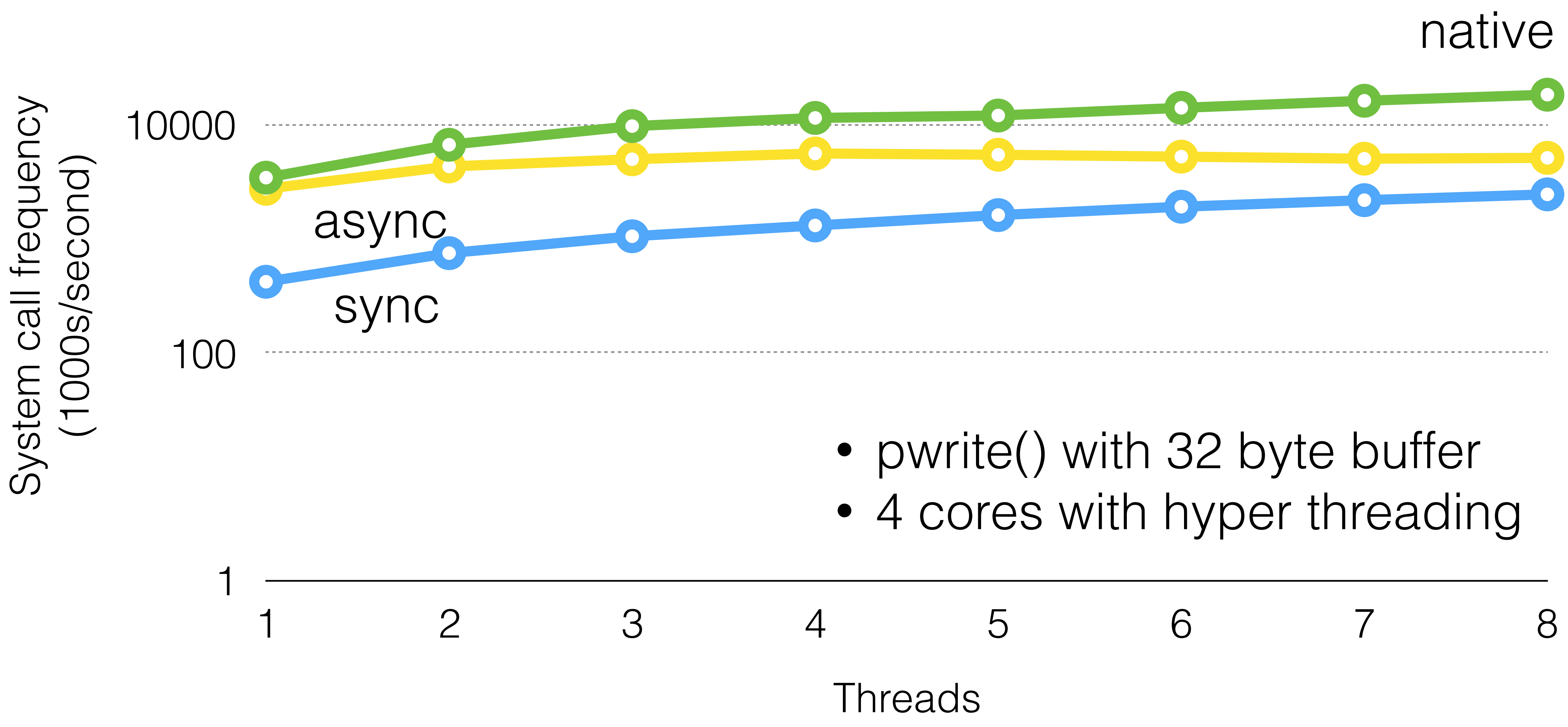
Container Integration



Container Integration

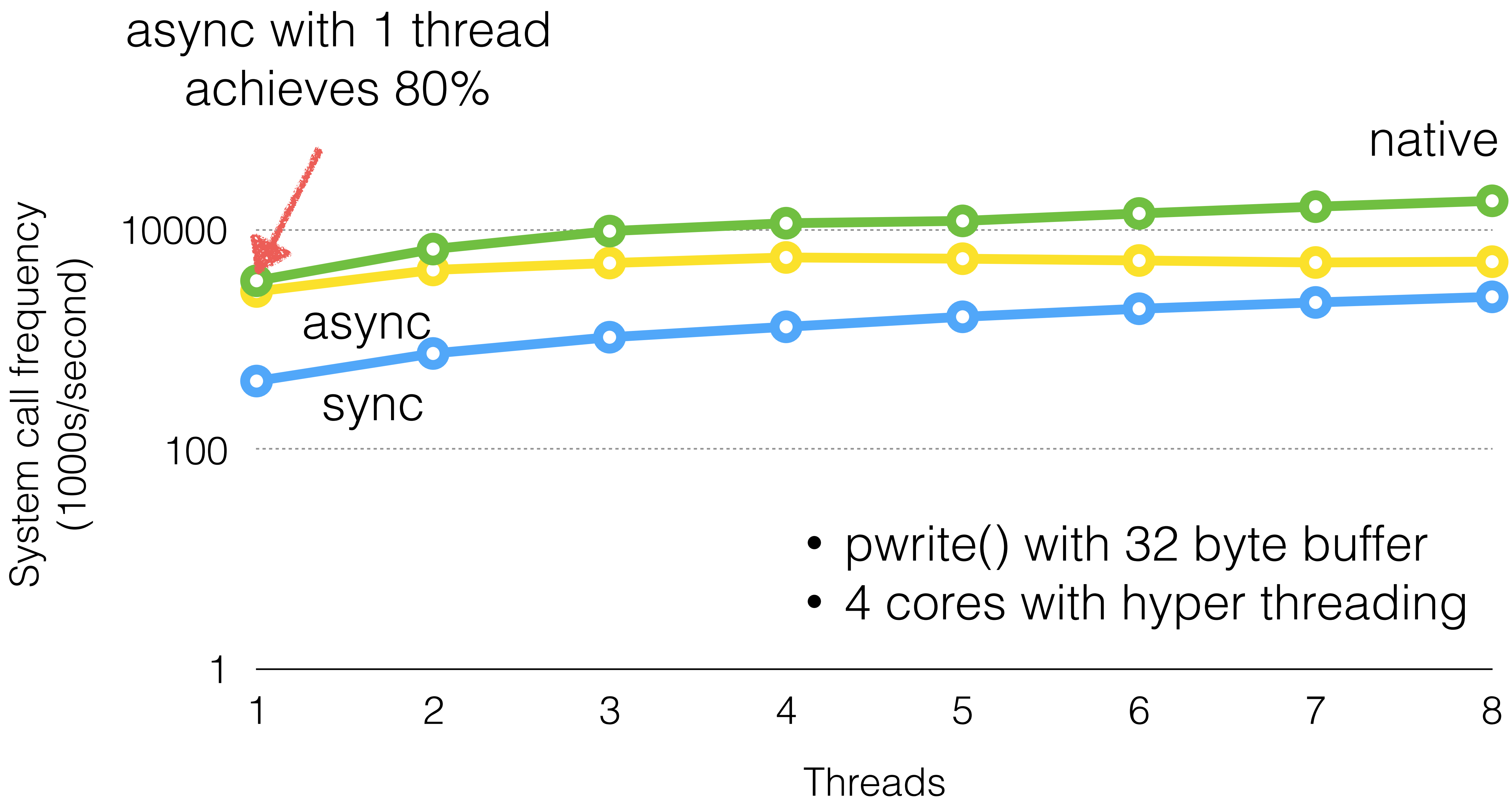


System Call Performance



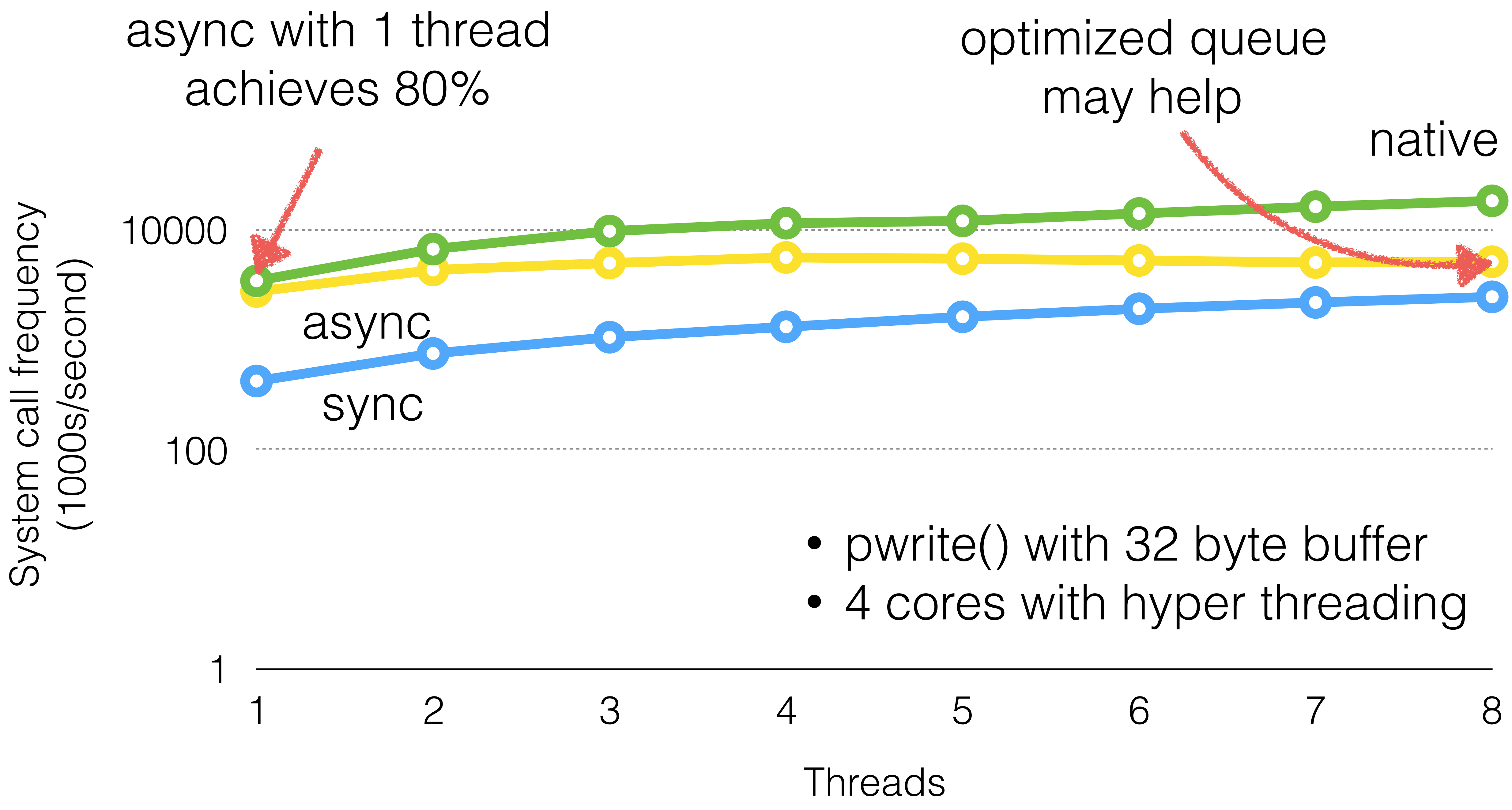
- pwrite() with 32 byte buffer
- 4 cores with hyper threading

System Call Performance



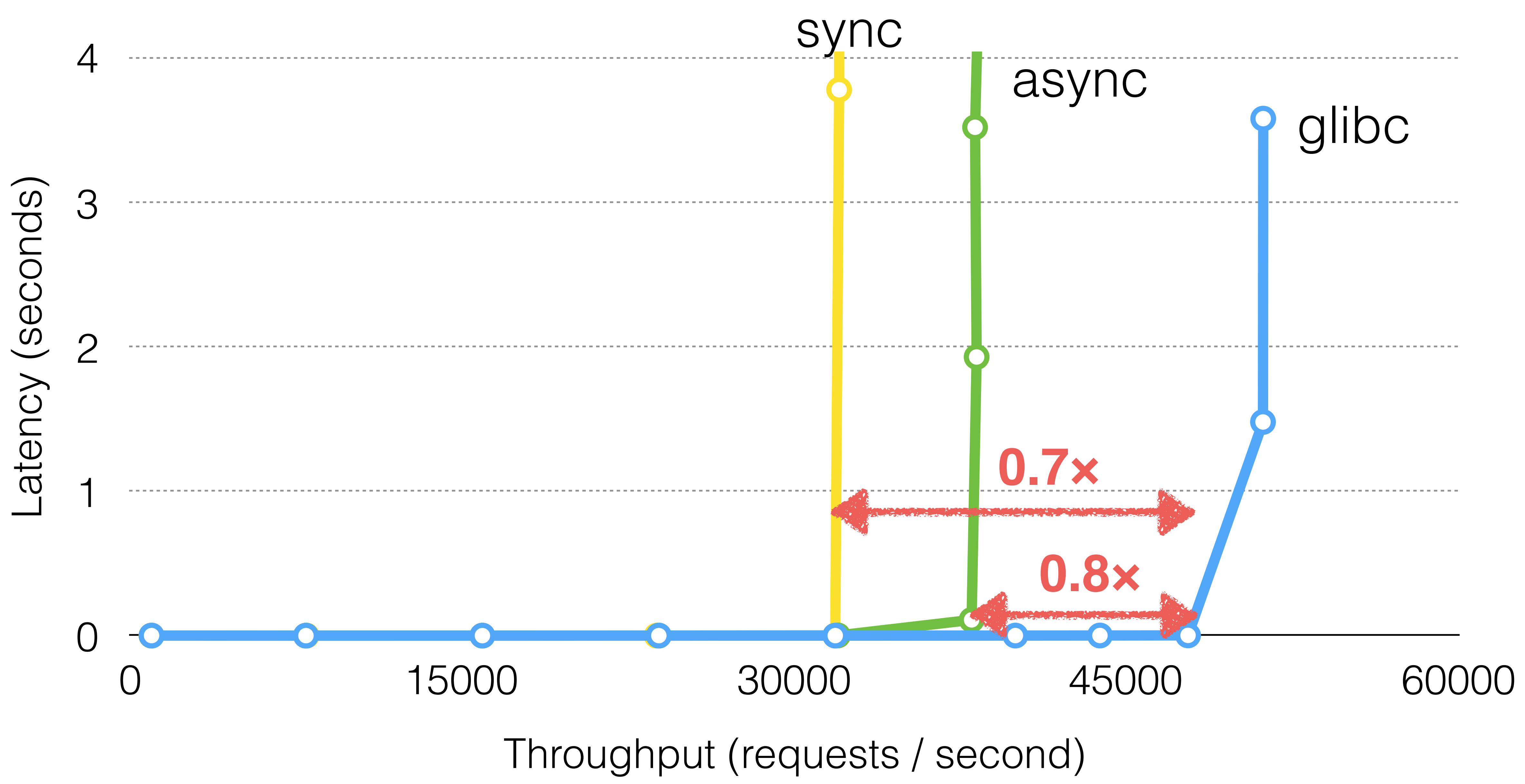
- pwrite() with 32 byte buffer
- 4 cores with hyper threading

System Call Performance



- pwrite() with 32 byte buffer
- 4 cores with hyper threading

Apache Throughput



Performance Overview

Application	Throughput w.r.t. native	
	async (%)	sync (%)
Memcached	120	113
Apache	80	70
NGINX	80	36
Redis	60	20

Performance Overview

Application	Throughput w.r.t. native	
	async (%)	sync (%)
Memcached	120	113
Apache	80	70
NGINX	80	36
Redis	60	20

inline encryption
has less overhead

Performance Overview

Application	Throughput w.r.t. native	
	async (%)	sync (%)
Memcached	120	113
Apache	80	70
NGINX	80	36
Redis	60	20

inline encryption has less overhead

inline encryption hurts performance with single thread

Summary

- Small trusted computing base (0.6× – 2.0× of native binary size)
- Low runtime overhead (0.6× – 1.2× of native throughput)
- Transparent to the container engine (e.g. Docker)