HybCache:
Hybrid Side-Channel-Resilient Caches for Trusted Execution Environments

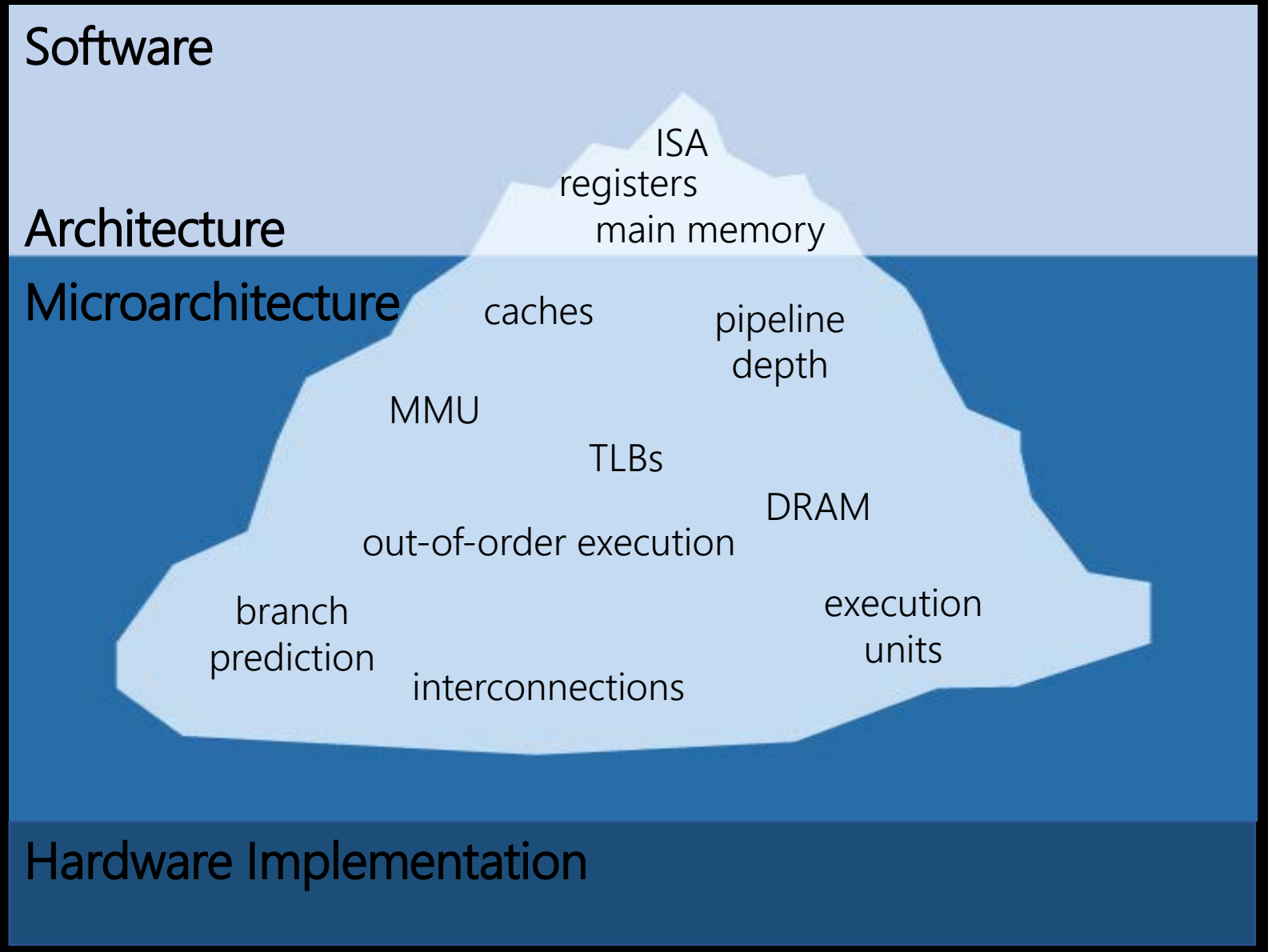
Ghada Dessouky, Tommaso Frassetto, Ahmad-Reza Sadeghi
Systems Security Lab, Technische Universität Darmstadt, Germany

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Architecture is Only the Tip of the Iceberg

The myth of the hardware-based trust anchor

Architecture is Only the Tip of the Iceberg



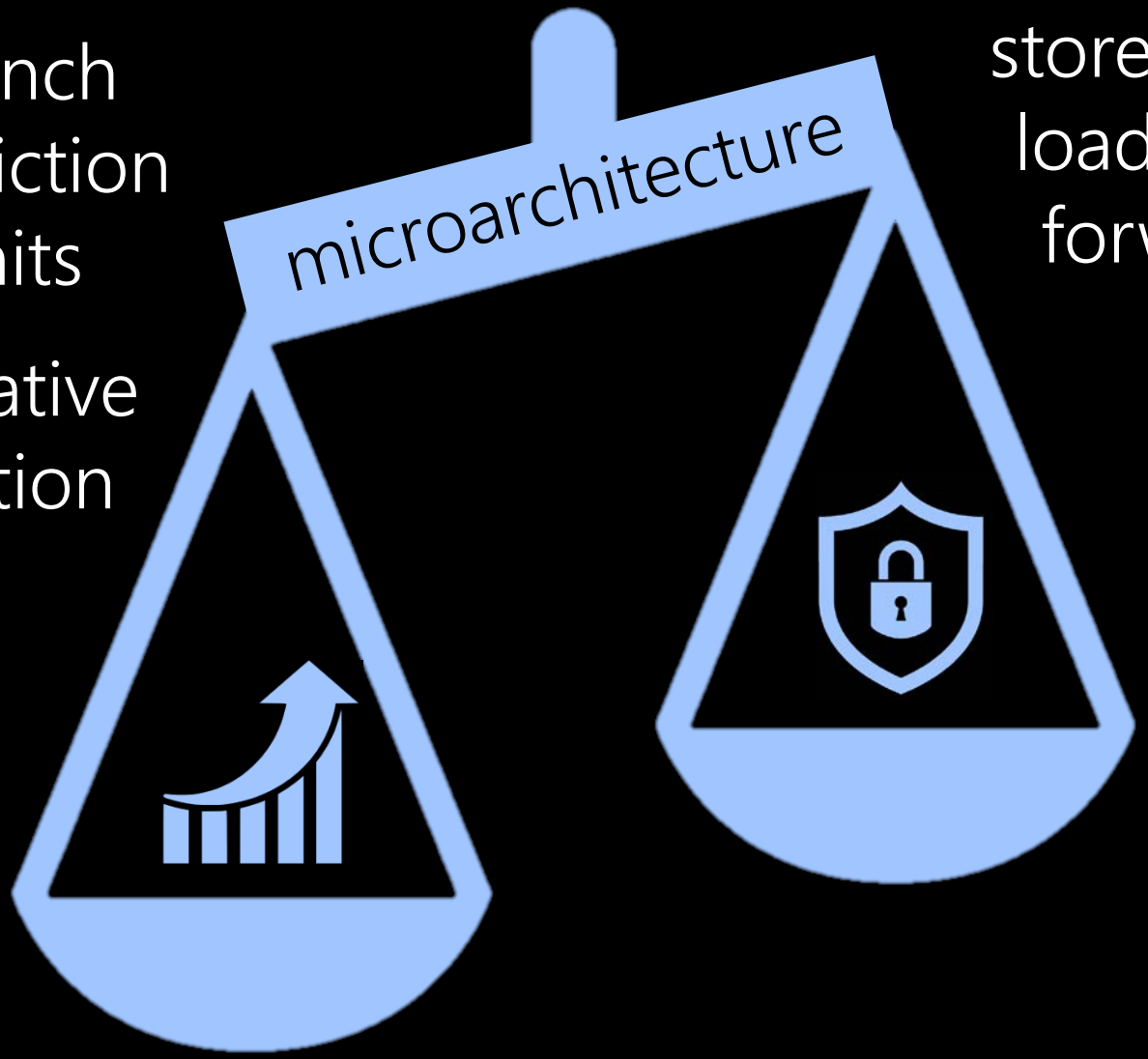
The Performance – Security Trade-Off

shared caches

branch prediction units

speculative execution

shared buffers contention

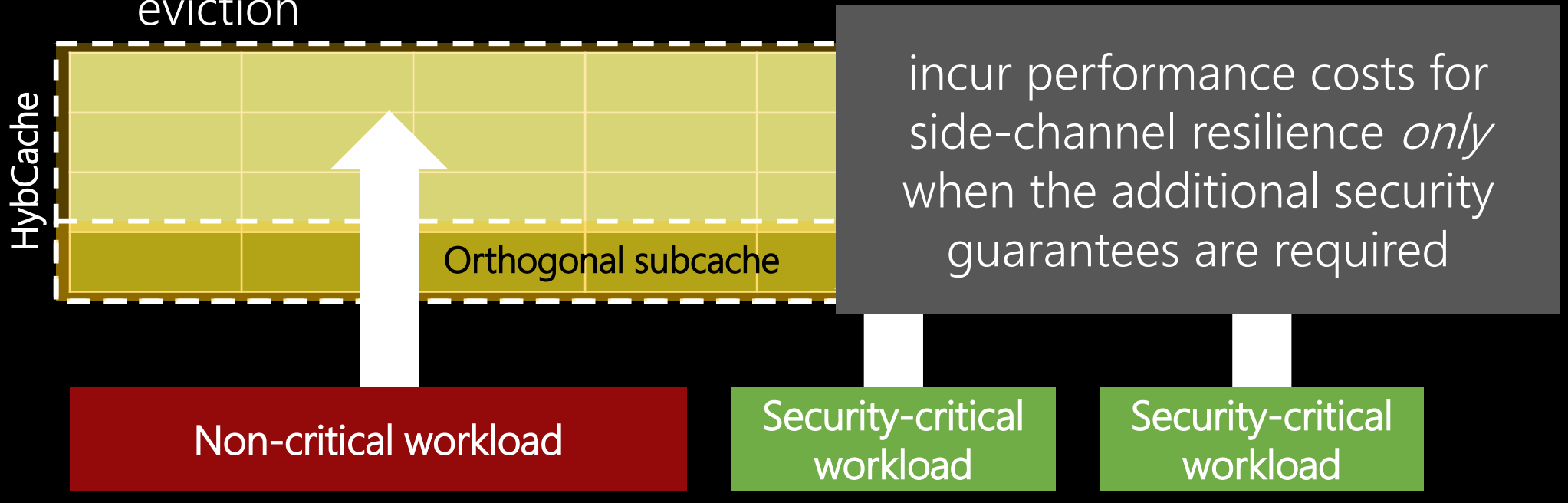


store-to-load/ load-to-load forwarding

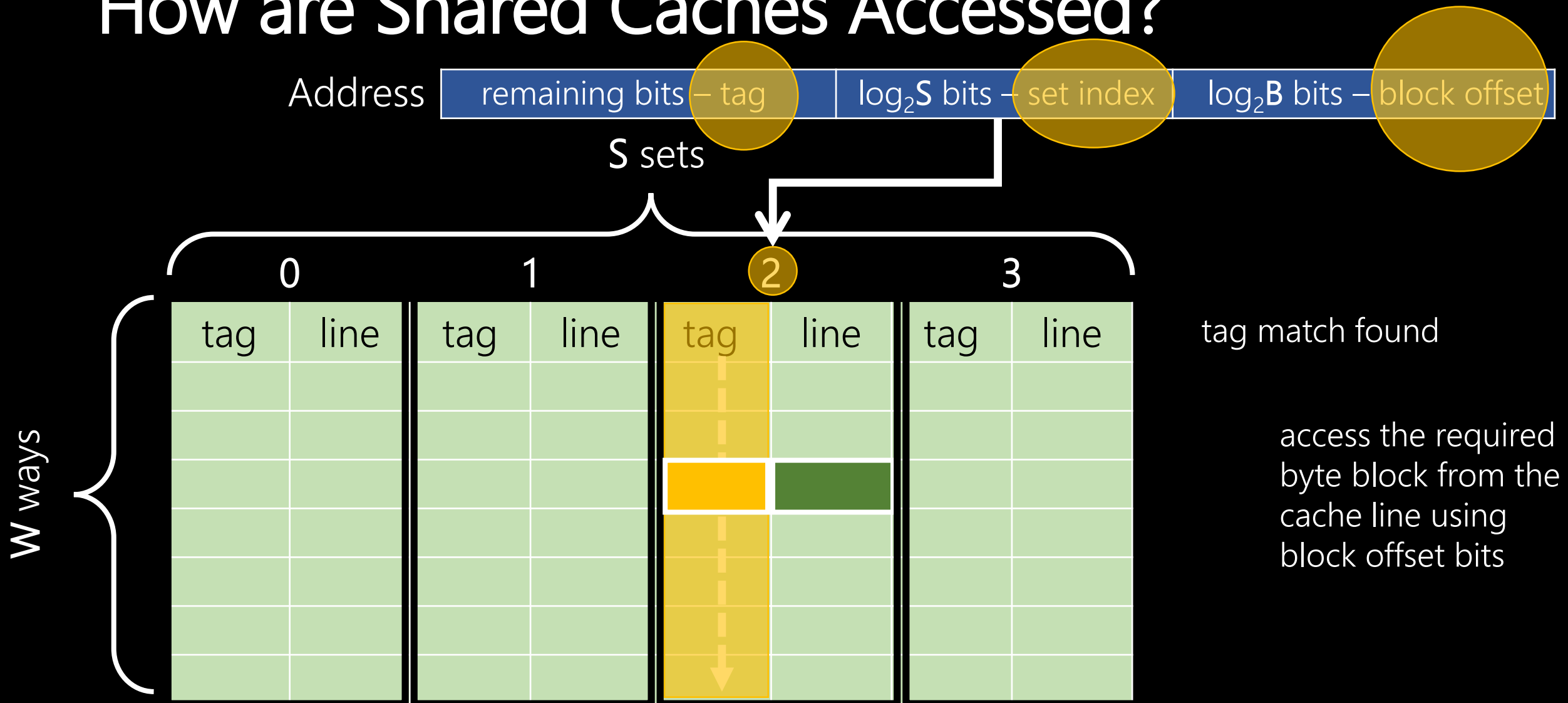
transient execution

HybCache: The Big Picture

- On-demand & configurable cache-based side-channel resilience → tunable performance/security trade-off knob
- Hybrid set-associative cache:
  - Non-critical workload: behaves typically and utilizes full cache capacity
  - Security-critical workload: utilizes only a cache subset fully-associatively with random eviction



How are Shared Caches Accessed?



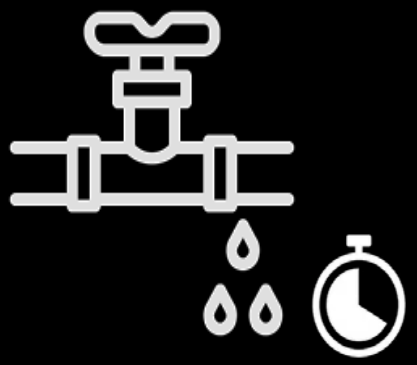
How are Shared Caches Exploited?

Because of the inherent principles of caches operation and their performance advantage

CPU (+cache) speed > DRAM access latency

- Cache miss: request data from DRAM upon first access → much slower
- Cache hit: No DRAM access required for subsequent accesses → much faster

inherent timing channel



Our Insights & Requirements for HybCache

Insights

- Majority of the execution workload is not security-critical
- Security-critical code is already isolated, e.g., in a TEE
- Root causes for conflict/contention & access-based cache attacks are set-associative eviction & shared cache lines
- Only approach to complete non-interference is strict cache partitioning → impractical

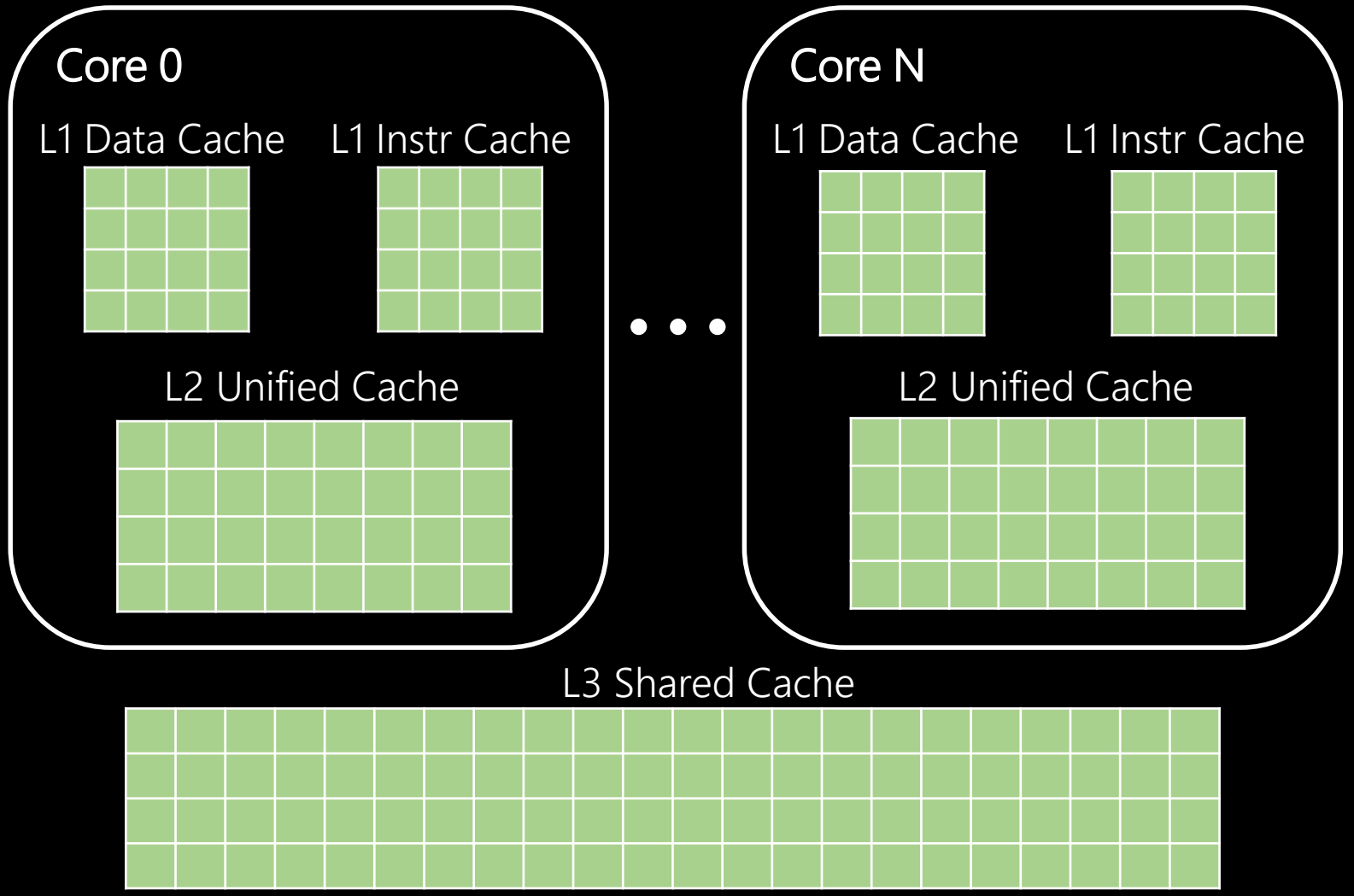
Requirements

- Strong side-channel resilience guarantees between security-critical and non-critical execution
- Side-channel resilience and dynamic utilization among mutually distrusting critical workloads
- Selective/configurable cache side-channel-resilience
- Usability and backwards compatibility

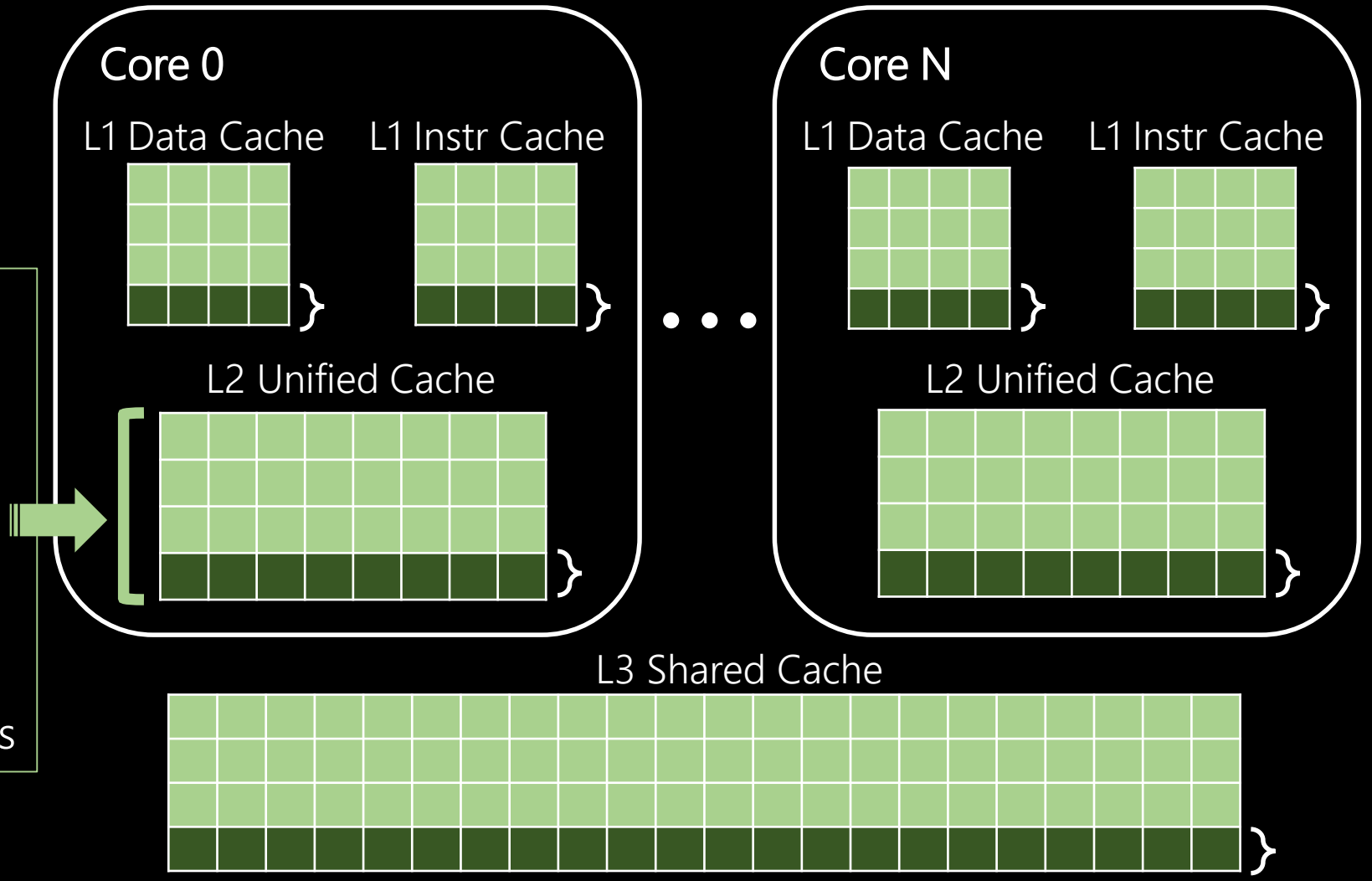
System Assumptions

- Security-critical code (requiring side-channel-resilient cache utilization) is in an isolated component or domain (I-Domains), e.g., a process or a TEE (enclave).
- Mutually distrusting code is allocated to different I-Domains.
- Isolated execution (in I-Domains) is the minority of the execution workload. Rest of the workload is non-isolated (NI-Domain).
- The attacker is not in the same I-Domain as the victim.

High-Level Architecture



High-Level Architecture

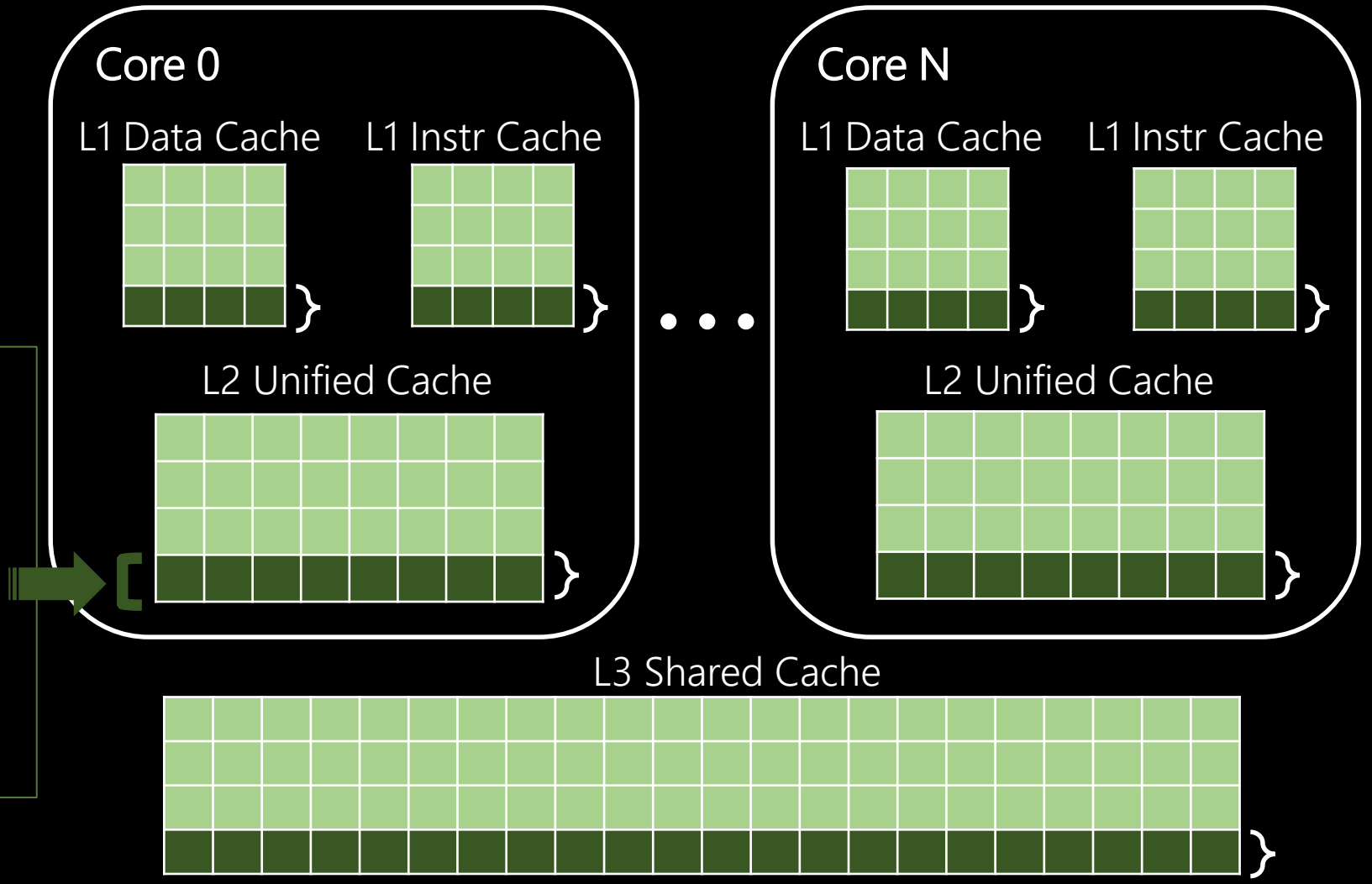


non-isolated cache access request from NI-Domain

full cache capacity including *subcache* ways

Orthogonal ***subcache*** of fully-associative cache ways

High-Level Architecture

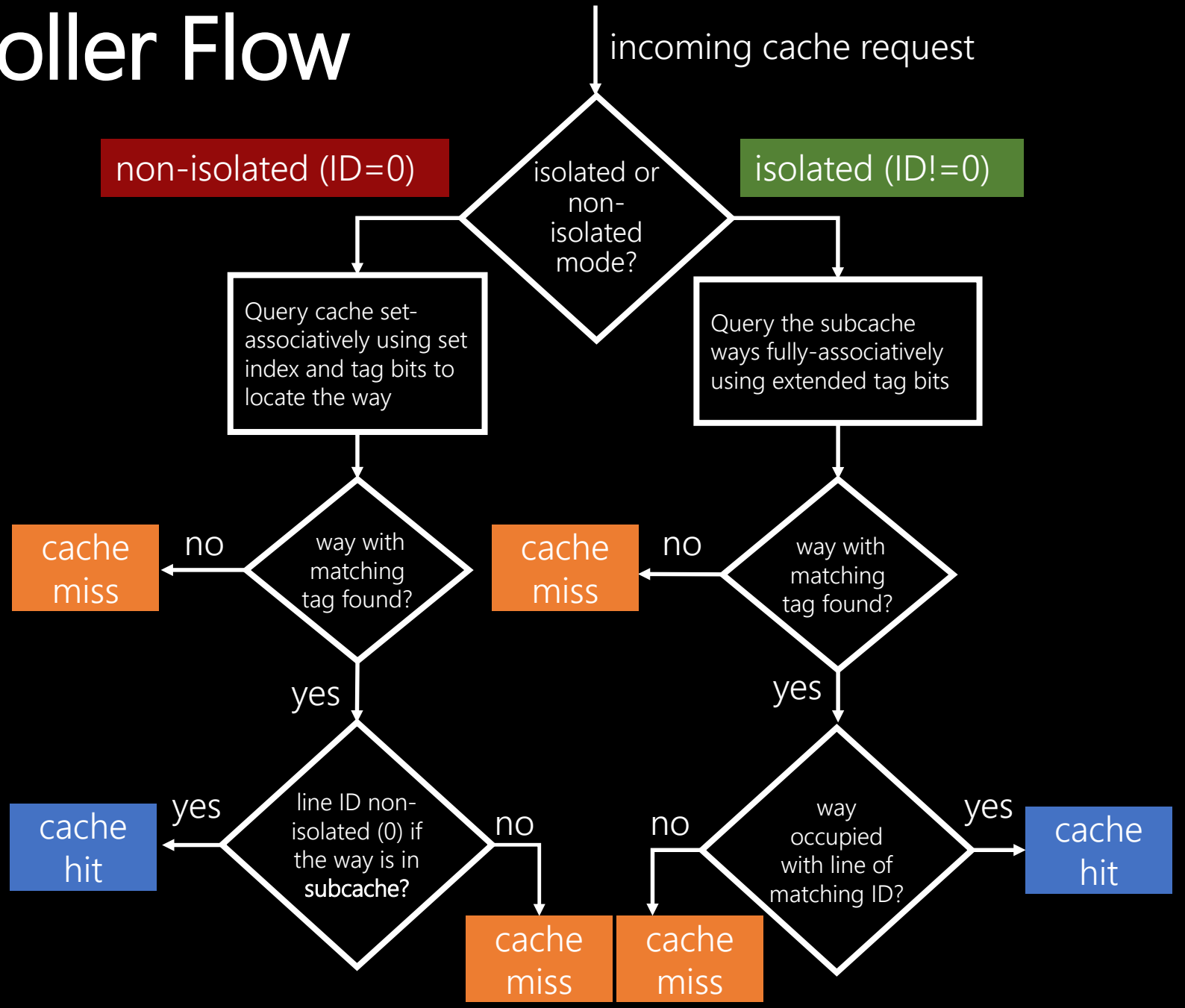


isolated cache access request (from any I-Domain):

restricted to only use *subcache* ways

Orthogonal ***subcache*** of fully-associative cache ways

Controller Flow



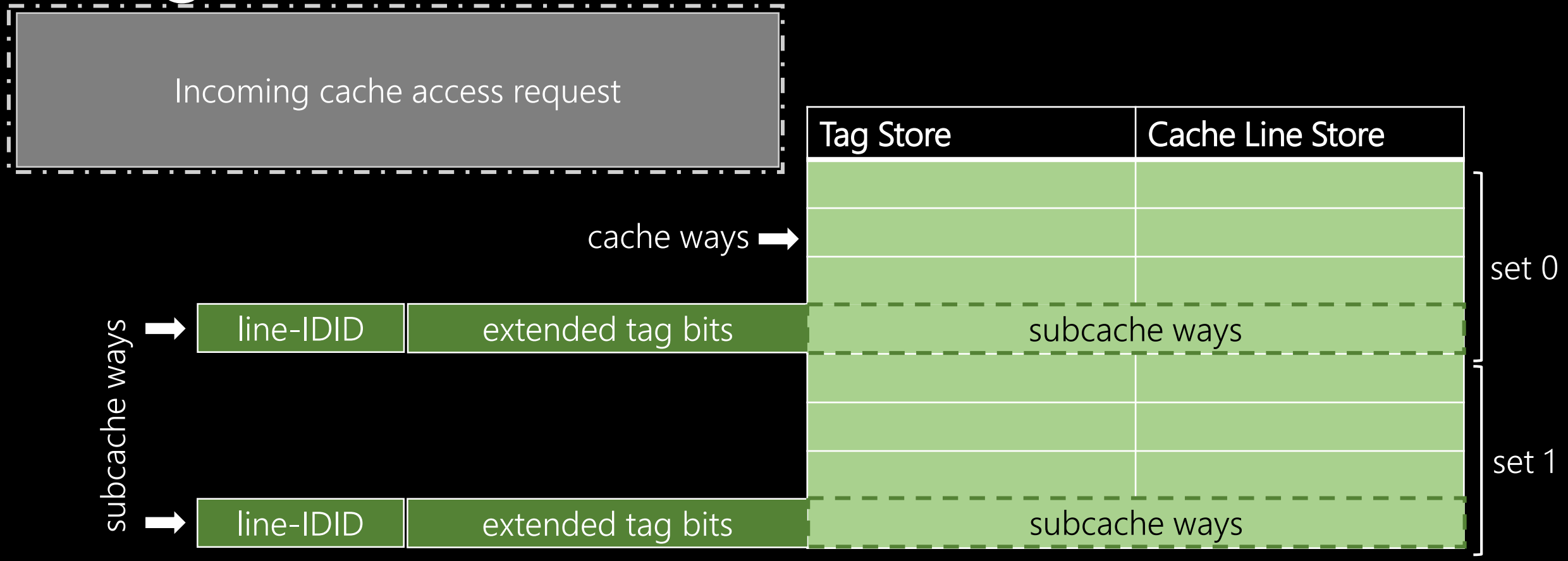
cache miss for non-isolated request

Evict and replace (via LRU/pseudo-LRU policy) cache line (including these occupying **subcache** ways)

cache miss for isolated request

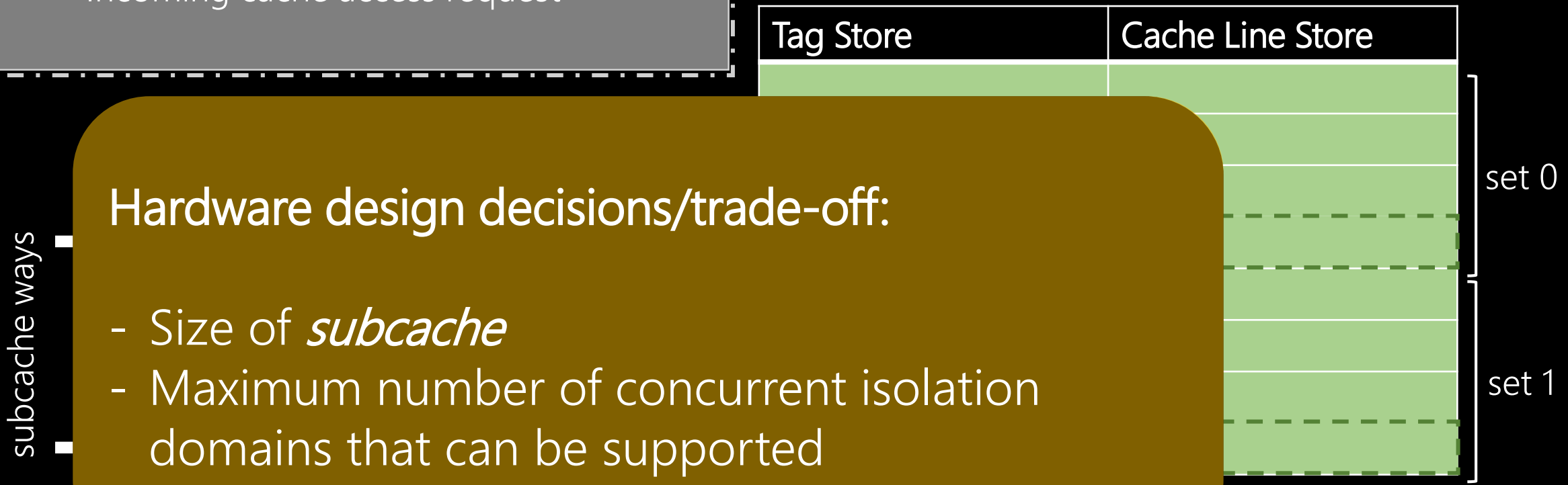
Randomly evict and replace any of the cache lines occupying the ***subcache*** ways (irrespective of their line-IDID)

Tag-Store Extensions



Tag-Store Extensions

Incoming cache access request



Hardware design decisions/trade-off:

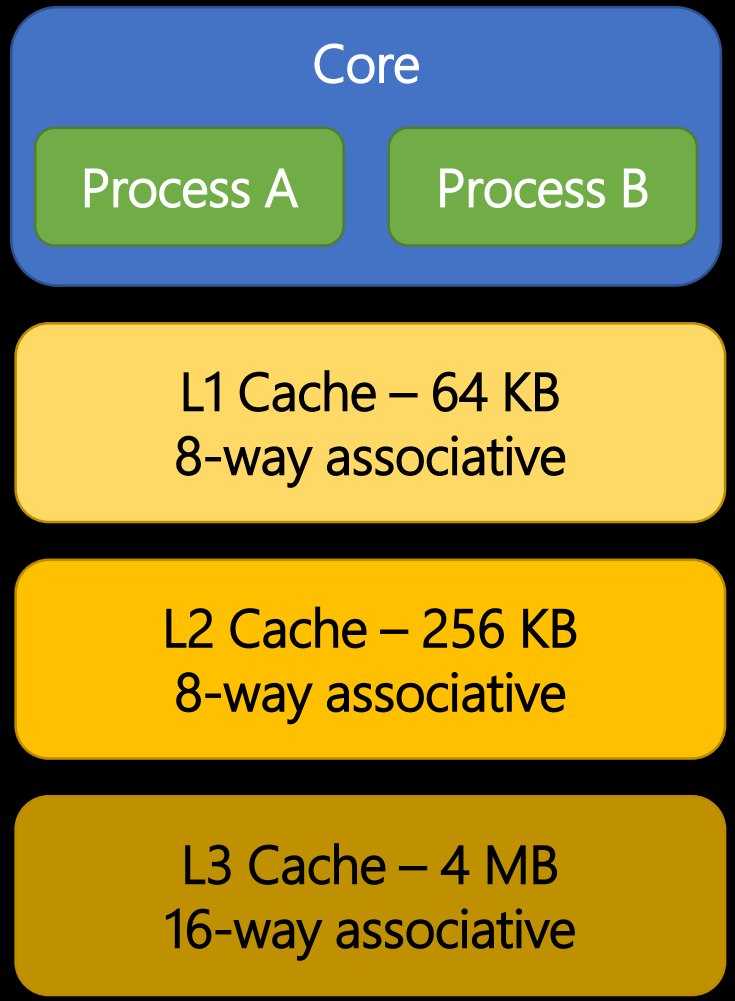
- Size of *subcache*
- Maximum number of concurrent isolation domains that can be supported

Security Analysis

Direct access to cache lines of another I-Domain not possible	• Shared cache lines are disallowed between different domains by design: cannot flush/hit on cache lines of another domain
Pre-computing and constructing an eviction set not possible	• Disabling the set-associative eviction of the subcache and random replacement → no reproducible and consistent mapping of a given memory access to a cache entry
Observing cache hits and misses of another I-Domain	• Cache hits not observable by another domain ✗ Attacker can still infer size of victim's working set by observing its own line evictions → cache occupancy channel remains by design

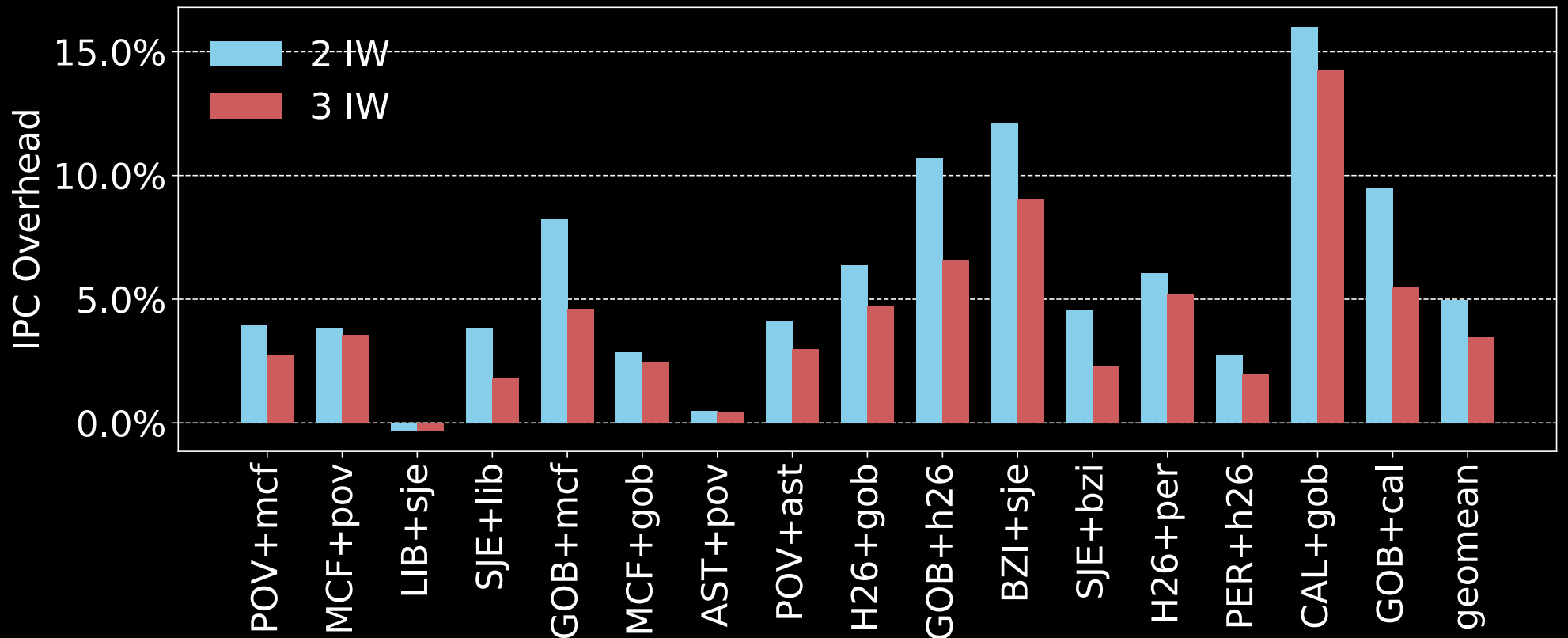
Evaluation

Performance: Two Processes

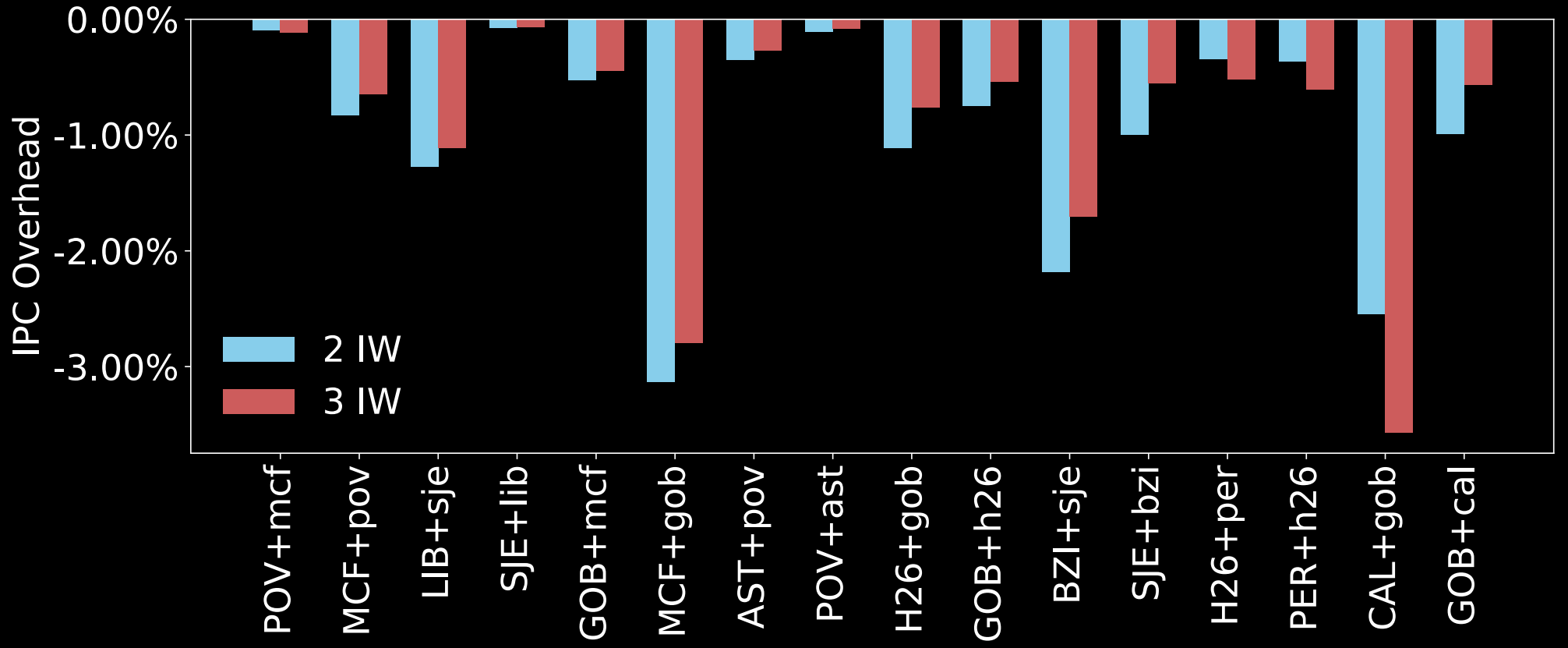


Mix	SPEC Benchmark
pov+mcf	povray, mcf
lib+sje	libquantum, sjeng
gob+mcf	gobmk, mcf
ast+pov	astar, povray
h26+gob	h264ref, gobmk
bzi+sje	bzip2, sjeng
h26+per	h264ref, perlbench
cal+gob	calculix, gobmk

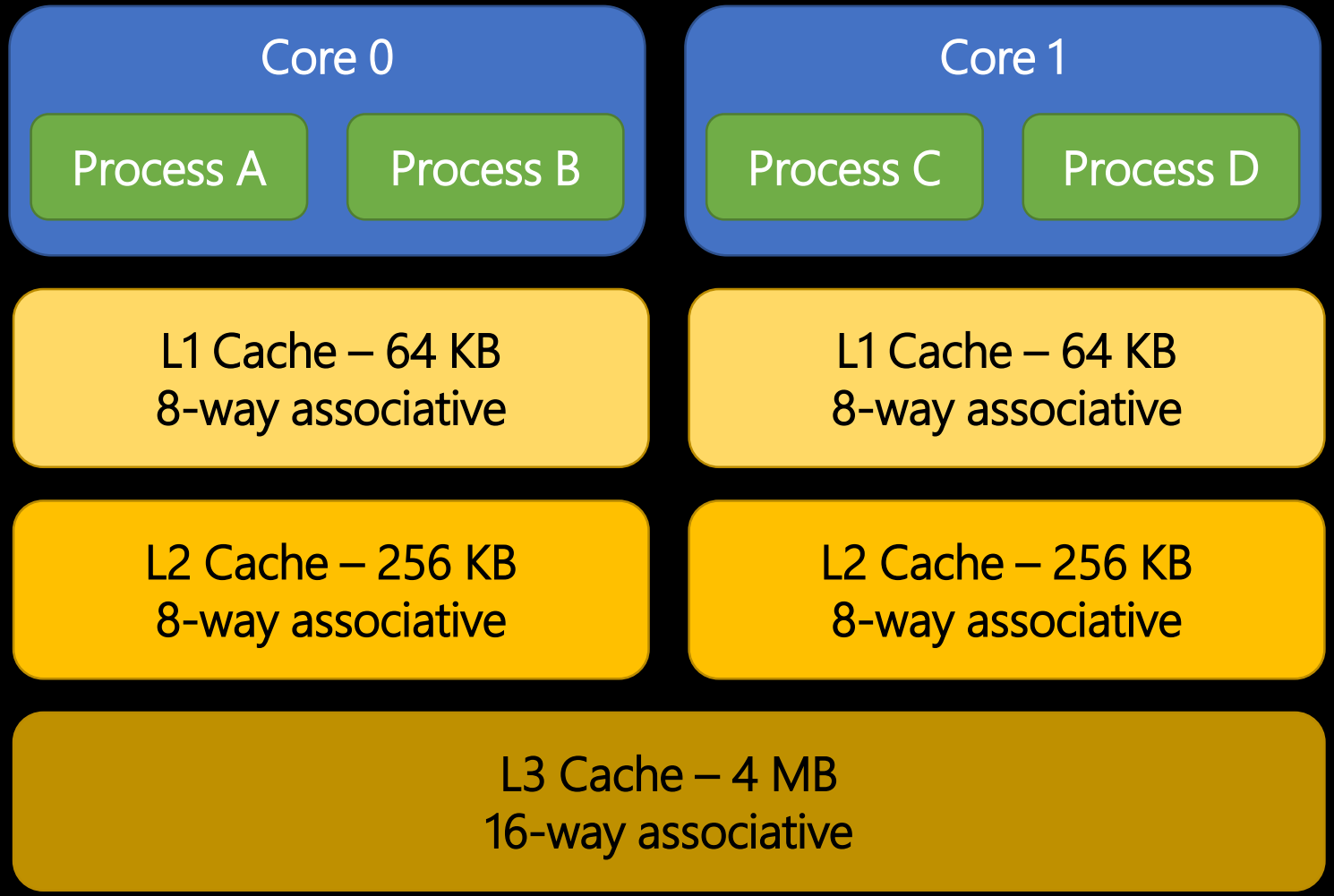
Performance Overhead of Isolated Processes



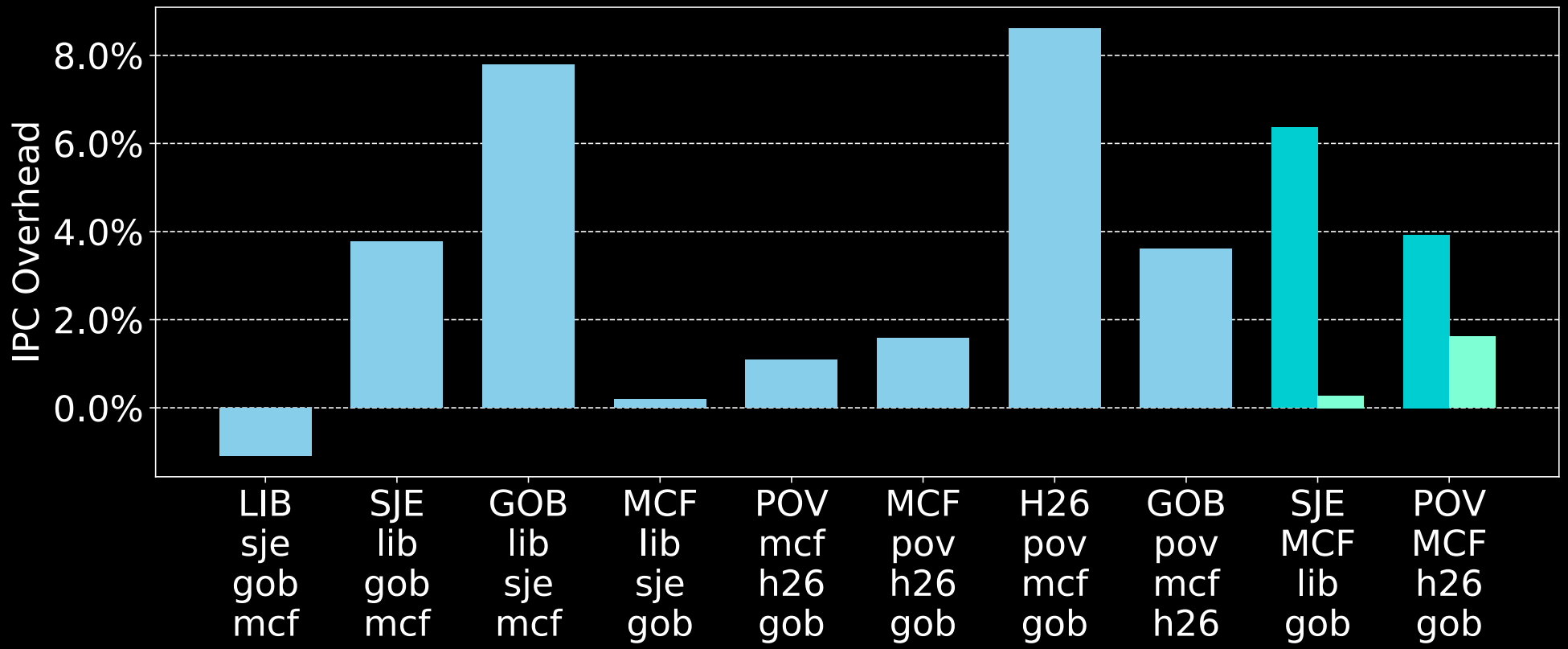
Overhead of Non-Isolated Processes



Performance: Four Processes



Performance Overhead of Isolated Processes



Deployment and Future Work

- Generic concept of “soft” partitioning:
  - which structures and caches to apply it to and the **subcache size** is a hardware design decision/trade-off
- CAM tag-store lookups are expensive (power consumption and timing/routing)
  - emerging memory technologies such as DRAM-based and STT-MRAM caches to alleviate the overheads
- Ongoing work: improved design to achieve the same full-associativity (+ stronger isolation) without expensive lookups

Thank you!

Ghada Dessouky
ghada.dessouky@trust.tu-darmstadt.de