

TEE-SHirT: Scalable Leakage-Free Cache Hierarchies for TEEs

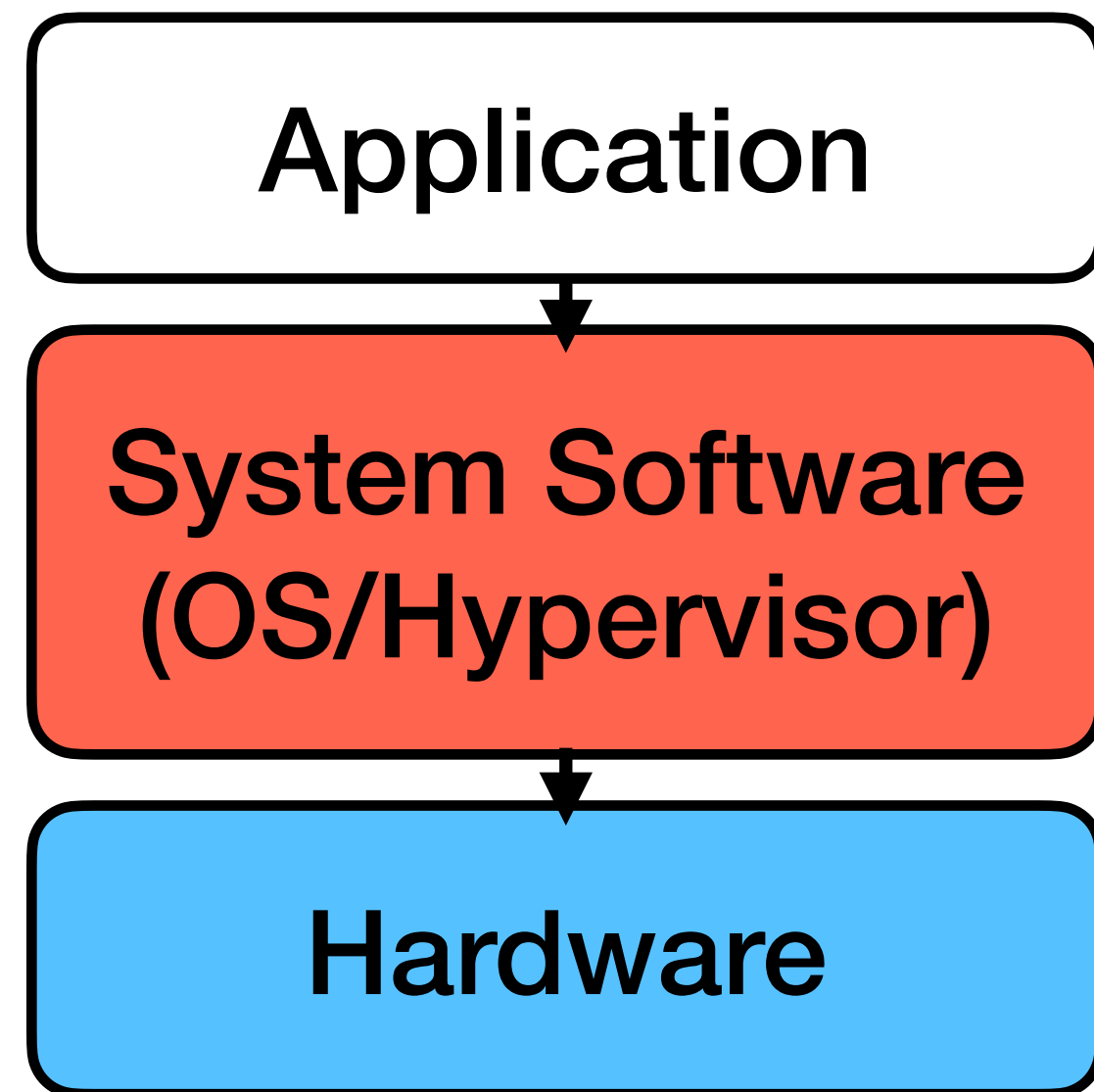
**Kerem Arıkan, Abraham Farrell, Williams Zhang Cen, Jack McMahon,
Barry Williams, Yu David Liu, Nael Abu-Ghazaleh*, Dmitry Ponomarev**

Binghamton University, *University of California, Riverside



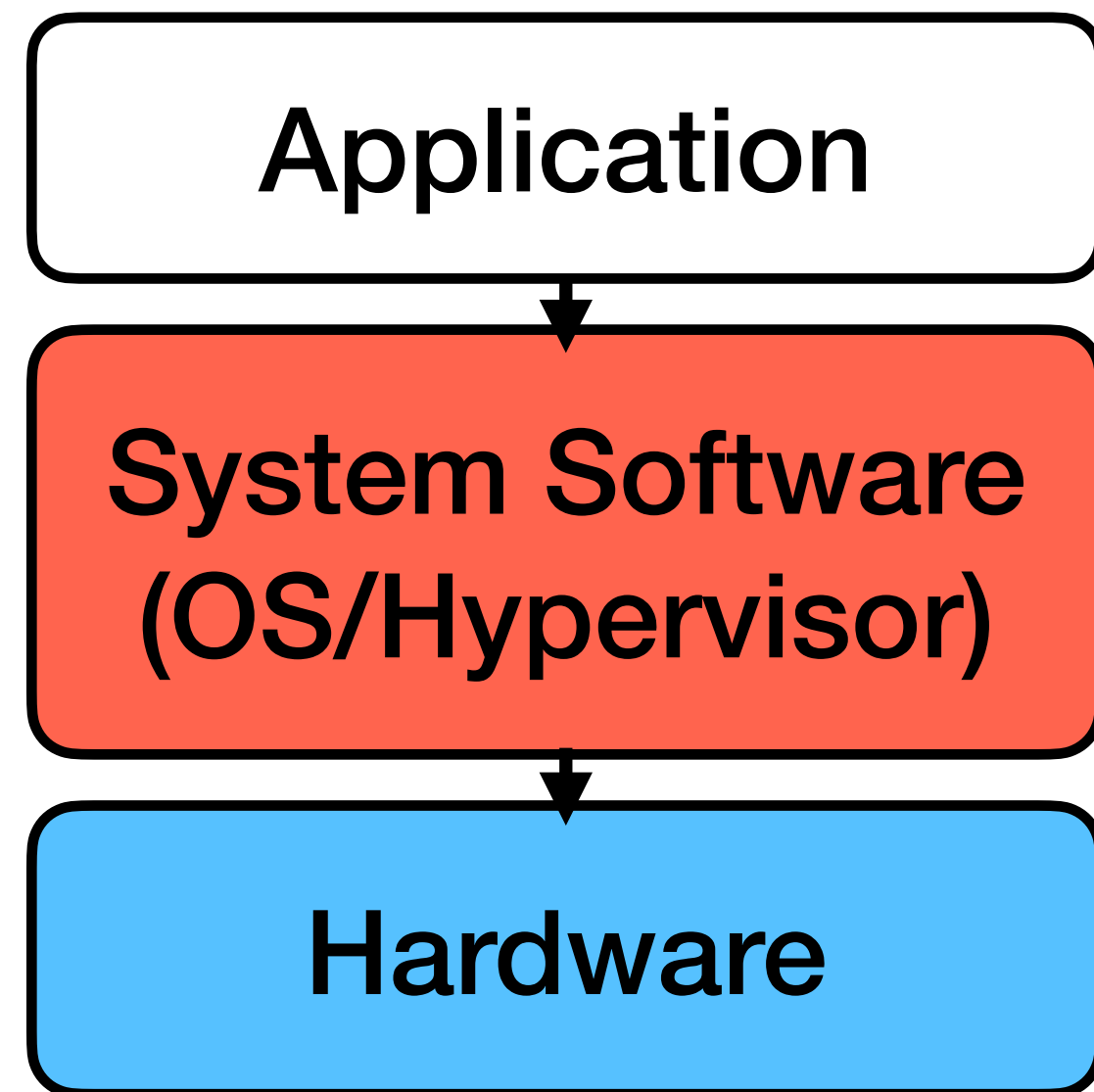
Trusted Execution Environment (TEE)

Traditional System Stack

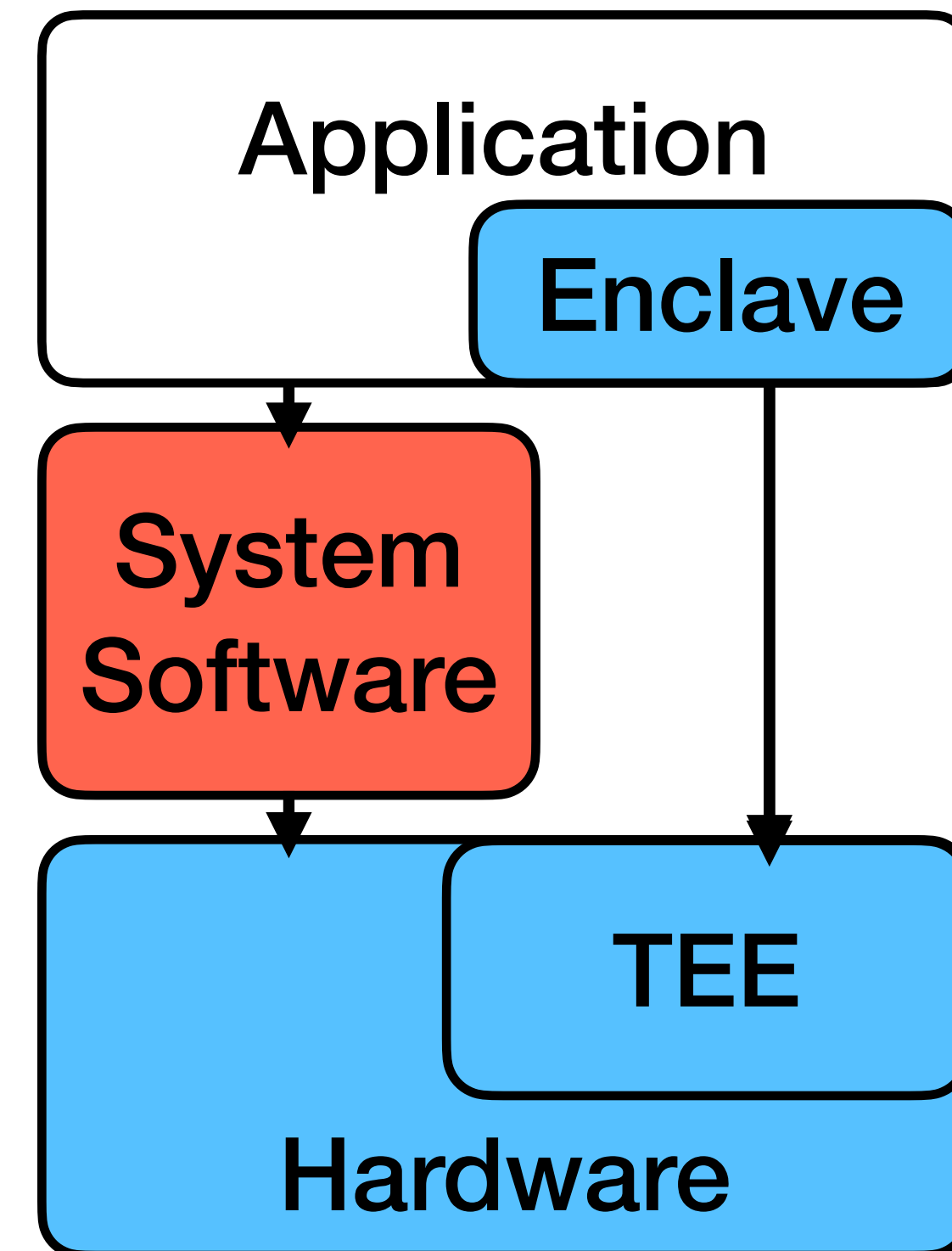


Trusted Execution Environment (TEE)

Traditional System Stack

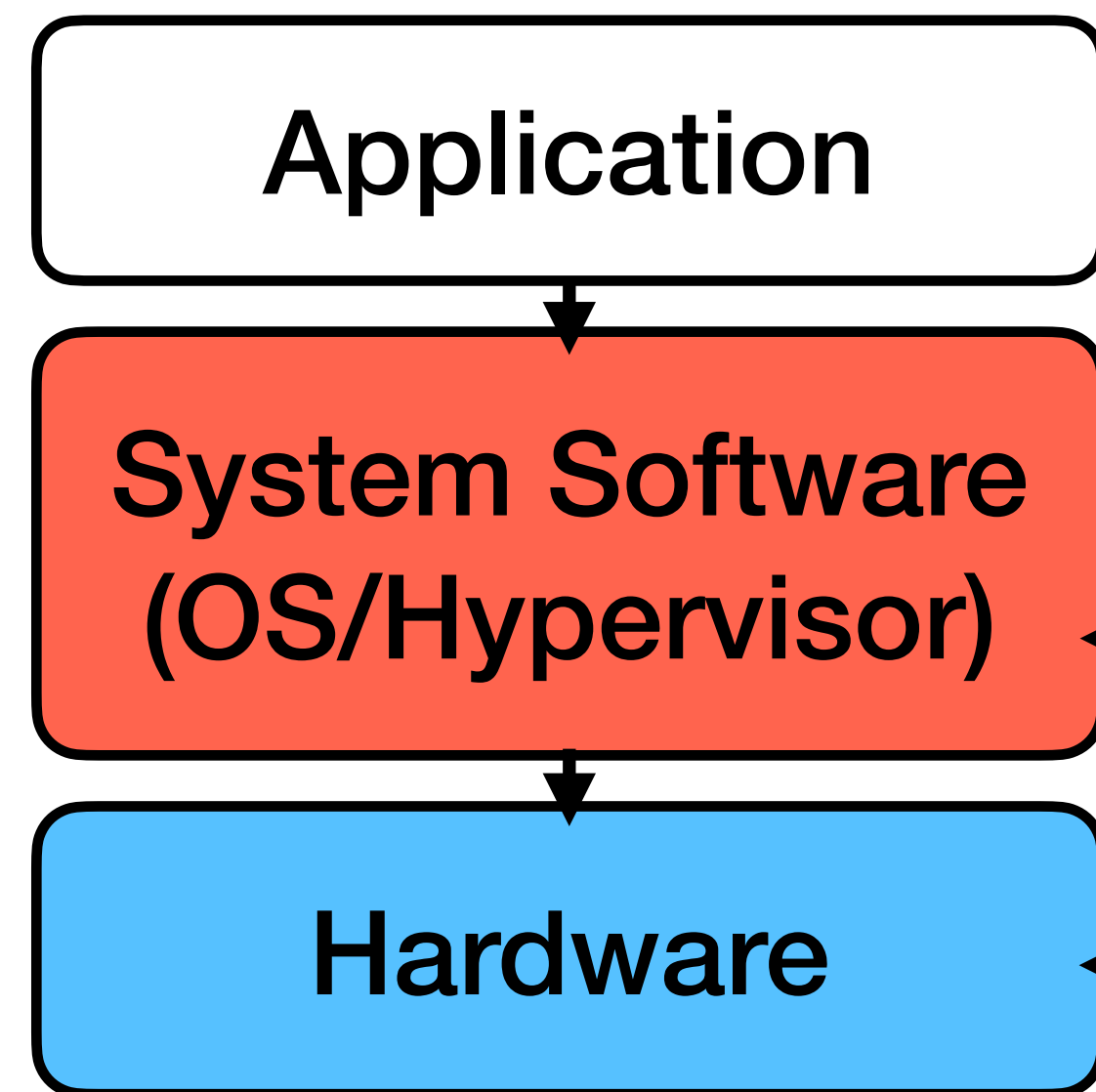


TEE System Stack

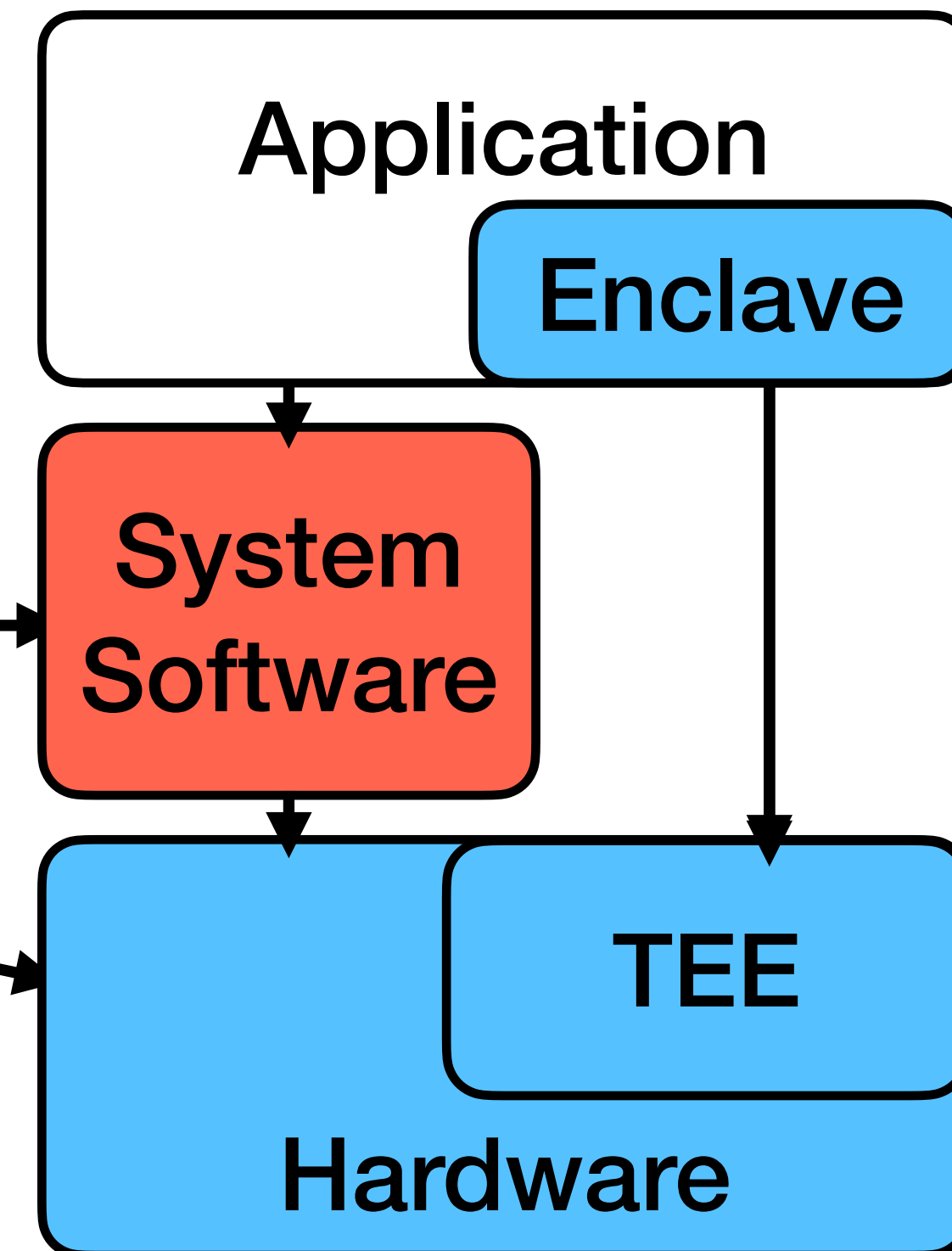


Trusted Execution Environment (TEE)

Traditional System Stack



TEE System Stack



Untrusted

Trusted

Can Information Still Leak in TEEs?



Intel patches up SGX best it can after another load of security holes found

Plus bugs squashed in Server Platform Services and more

By Dan Robinson

Wed 15 Feb 2023 | 20:40 UTC

Intel's Software Guard Extensions (SGX) are under the spotlight again after the chipmaker disclosed several newly discovered vulnerabilities affecting the tech, and recommended users update their firmware.

The security holes are among the latest disclosures listed on Intel's [Security Center](#) page. These cover a wide range of Intel products including Xeon processors, network adapters, and also software.

Overall, there were 31 advisories added to the Intel Security Center as of February 14, as we noted [here](#). There were five CVE-listed SGX-related security holes tackled in that Patch Tuesday patch.

Two of the SGX vulnerabilities involve potential escalation of privilege that could lead to information disclosure, which is awkward for a feature that is supposed to enable secure processing of sensitive data inside encrypted memory areas known as enclaves.

One, [CVE-2022-38090](#), has a severity rating of medium and affects a number of Intel processors, including the 3rd Gen Xeon Scalable server chips, which have only recently



Home > Security > New CPU attack technique can leak secrets from Intel SGX enclaves

By Luctor Constantin

CSO Senior Writer

New CPU attack technique can leak secrets from Intel SGX enclaves

News Analysis

Mar 10, 2023 | 7 mins

Security | Vulnerabilities

The Load Value Injection attack can bypass security boundaries and mitigations put in place for other CPU vulnerabilities such as Spectre and Meltdown.



Credit: Intel | Baks / Getty Images

Researchers have devised a new attack against Intel CPUs that can leak sensitive secrets stored in SGX secure enclaves and, at least in theory, from privileged processes across security boundaries such as kernel



US Edition

Home | Reviews | Best Picks | Raspberry Pi | CPUs | GPUs | Coupons

TRENDING | Intel Core i9-14900K | AMD RX 7800 XT | Intel Arc A580

When you purchase through links on our site, we may earn an affiliate commission. [Here's how it works.](#)

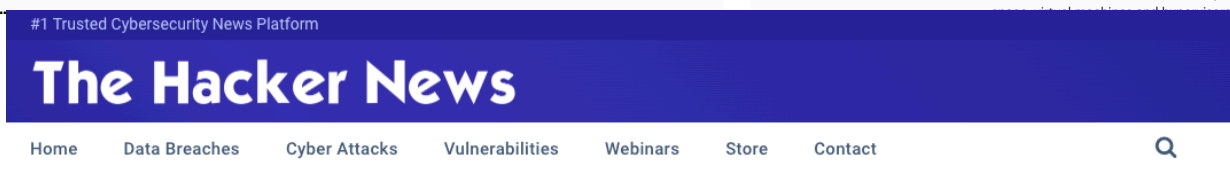
News

Intel 'Sunny Cove' SGX Vulnerability Discovered

By Anton Shilov published August 10, 2022

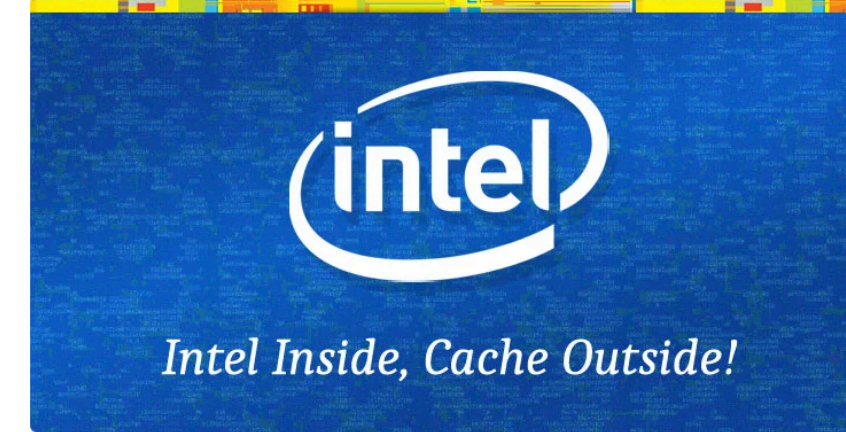
Yet another hardware bug affects Intel SGX-enabled CPUs.

Comments (2)



New 'CacheOut' Attack Leaks Data from Intel CPUs, VMs and SGX Enclave

Jan 28, 2020 | Mohit Kumar



Another month, another speculative execution vulnerability found in Intel processors. If your computer is running any modern Intel CPU built before October 2018, it's likely vulnerable to a newly discovered hardware issue that could allow attackers to leak sensitive data from the OS kernel, co-resident virtual machines, and even from Intel's secured SGX enclave.

- Researchers Uncover Undetectable Crypto Mining Technique on Azure Automation
- StripedFly Malware Operated Unnoticed for 5 Years, Infecting 1 Million Devices
- Researchers Find 34 Windows Drivers Vulnerable to Full Device Takeover
- CISA Alerts: High-Severity SLP Vulnerability Now Under Active Exploitation
- 48 Malicious npm Packages Found Deploying Reverse Shells on Developer Systems
- CanesSpy Spyware Discovered in Modified WhatsApp Versions



New SGAXe attack steals protected data from Intel SGX enclaves

By Sergiu Gatlan

June 9, 2020 | 01:00 PM



Intel processors are vulnerable to a new attack known as SGAXe that breaches the security guarantees of Intel Software Guard eXtensions (SGX) enclaves. It is designed to specifically target and leak data from Intel processors.



Can Information Still Leak in TEEs?

The Register

Intel patches up SGX best it can after another load of security holes found

Plus bugs squashed in Server Platform Services and more

By Dan Robinson | Wed 15 Feb 2023 | 20:40 UTC

Intel's Software Guard Extensions (SGX) are under the spotlight again after the chipmaker disclosed several newly discovered vulnerabilities affecting the tech, and recommended users update their firmware.

The security holes are among the latest disclosures listed on Intel's [Security Center](#) page. These cover a wide range of Intel products including Xeon processors, network adapters, and also software.


Overall, there were 31 advisories added to the Intel Security Center as of February 14.

CSO

New CPU attack technique can leak secrets from Intel SGX enclaves

By Lucian Constantin | Mar 10, 2022 | 7 mins

The Load Value Injection attack can bypass security boundaries and mitigations put in place for other CPU vulnerabilities such as Spectre and Meltdown.



tom's HARDWARE

US Edition

TRENDING: Intel Core i9-14900K, AMD RX 7800 XT, Intel Arc A580

Intel 'Sunny Cove' SGX Vulnerability Discovered

By Anton Shilov published August 10, 2022

Yet another hardware bug affects Intel SGX-enabled CPUs.

Comments (2)

 **Side-Channel Attacks!** 

The Hacker News

New 'CacheOut' Attack Leaks Data from Intel CPUs, VMs and SGX Enclave

By Jan 28, 2020 | Mohit Kumar



Another month, another speculative execution vulnerability found in Intel processors.

If your computer is running any modern Intel CPU built before October 2018, it's likely vulnerable to a newly discovered hardware issue that could allow attackers to leak sensitive data from the OS kernel, co-resident virtual machines, and even from Intel's secured SGX enclave.

Trending News

- Researchers Uncover Undetectable Crypto Mining Technique on Azure Automation
- StripedFly Malware Operated Unnoticed for 5 Years, Infecting 1 Million Devices
- Researchers Find 34 Windows Drivers Vulnerable to Full Device Takeover
- CISA Alerts: High-Severity SLP Vulnerability Now Under Active Exploitation
- 48 Malicious npm Packages Found Deploying Reverse Shells on Developer Systems
- Canespy Spyware Discovered in Modified WhatsApp Versions

DEEPI INDCOMPUTER

New SGAXe attack steals protected data from Intel SGX enclaves

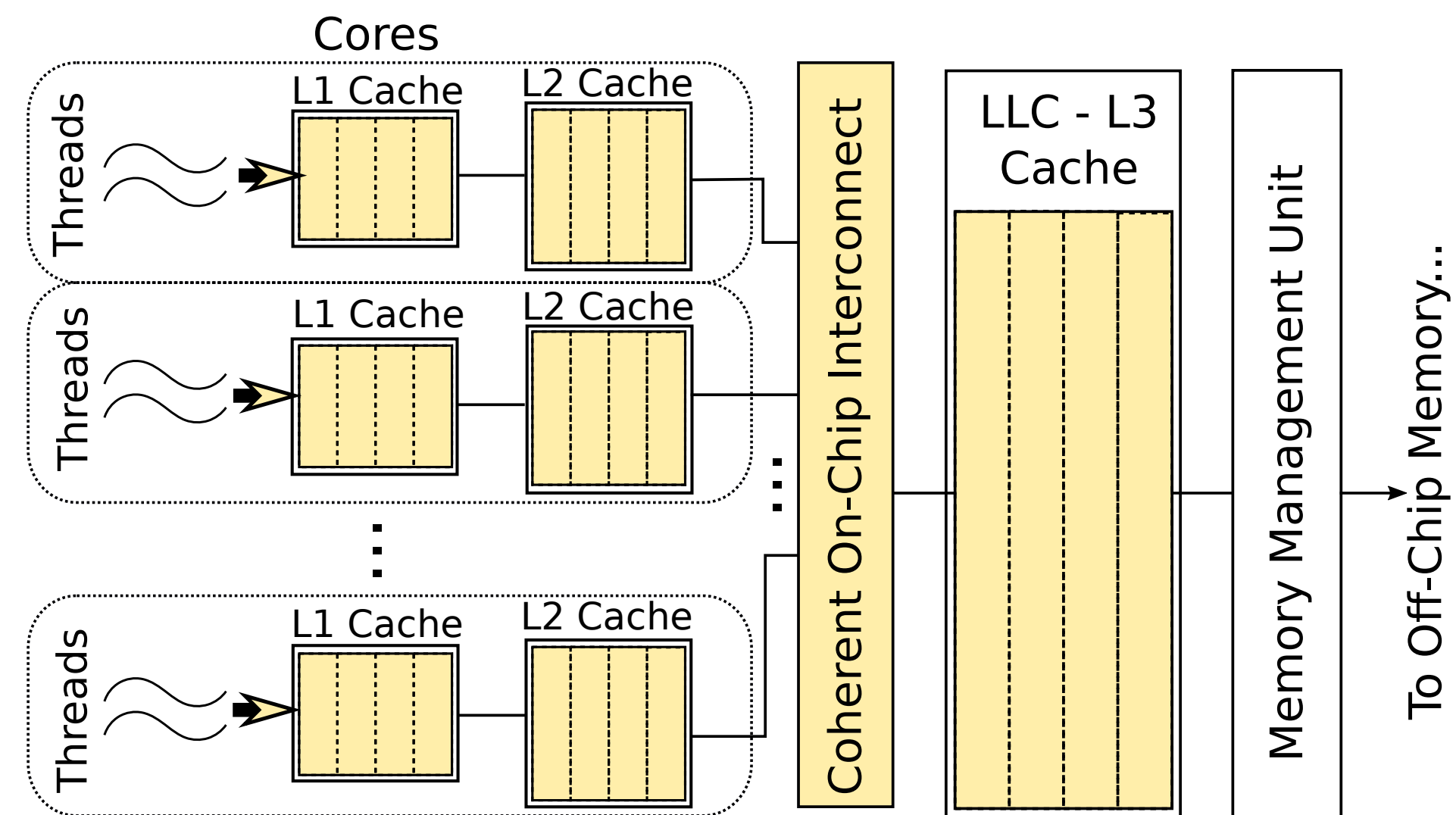
By Sergiu Gatlan | June 9, 2020 | 01:00 PM



Intel processors are vulnerable to a new attack known as SGAXe that breaches the security guarantees of Intel Software Guard eXtensions (SGX) enclaves. It is designed to specifically target and leak data from Intel processors.

Side-Channel Attacks on Caches

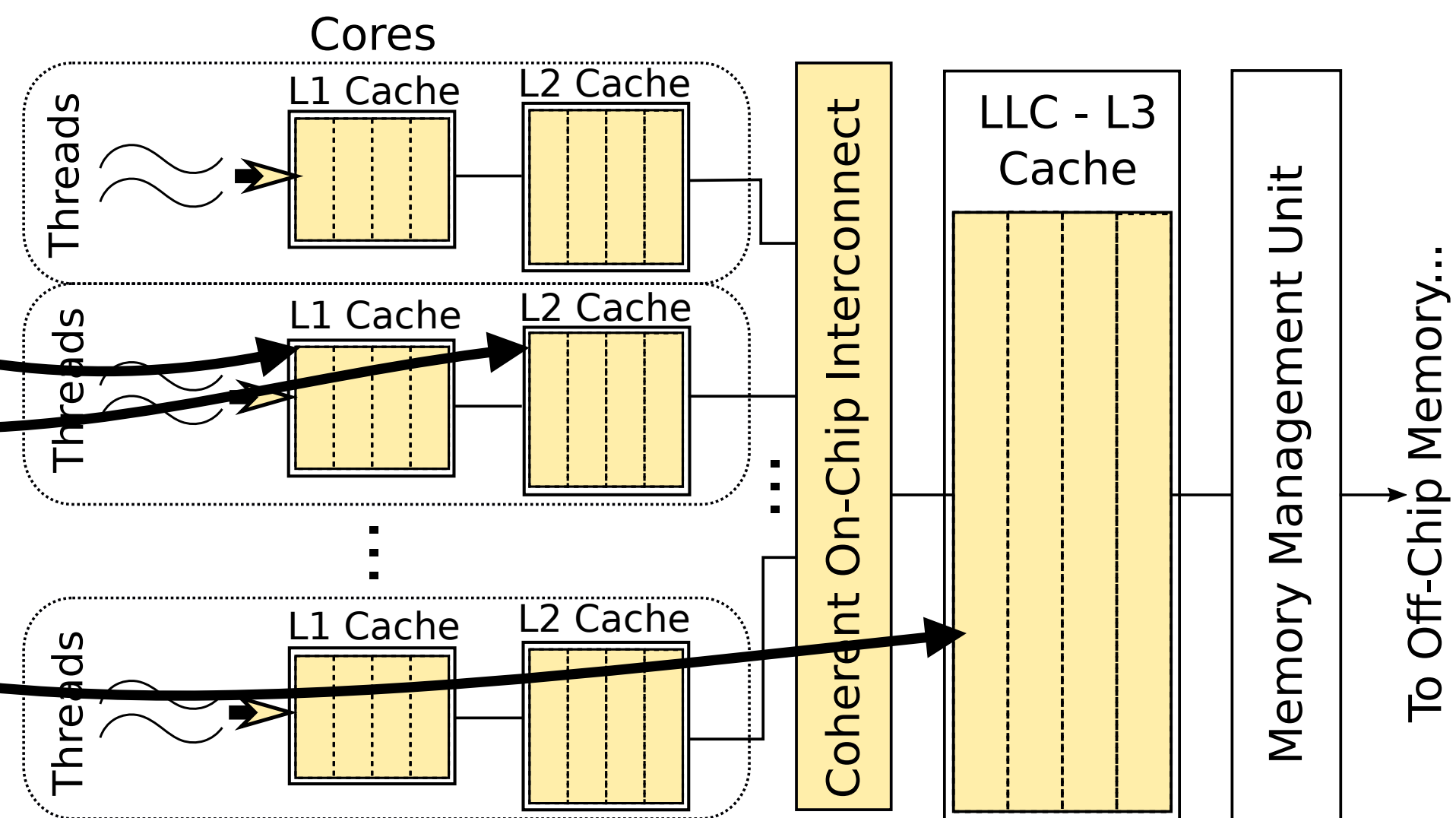
- All cache levels are vulnerable
- The system software can be a part of an attack



Side-Channel Attacks on Caches

All cache levels are vulnerable

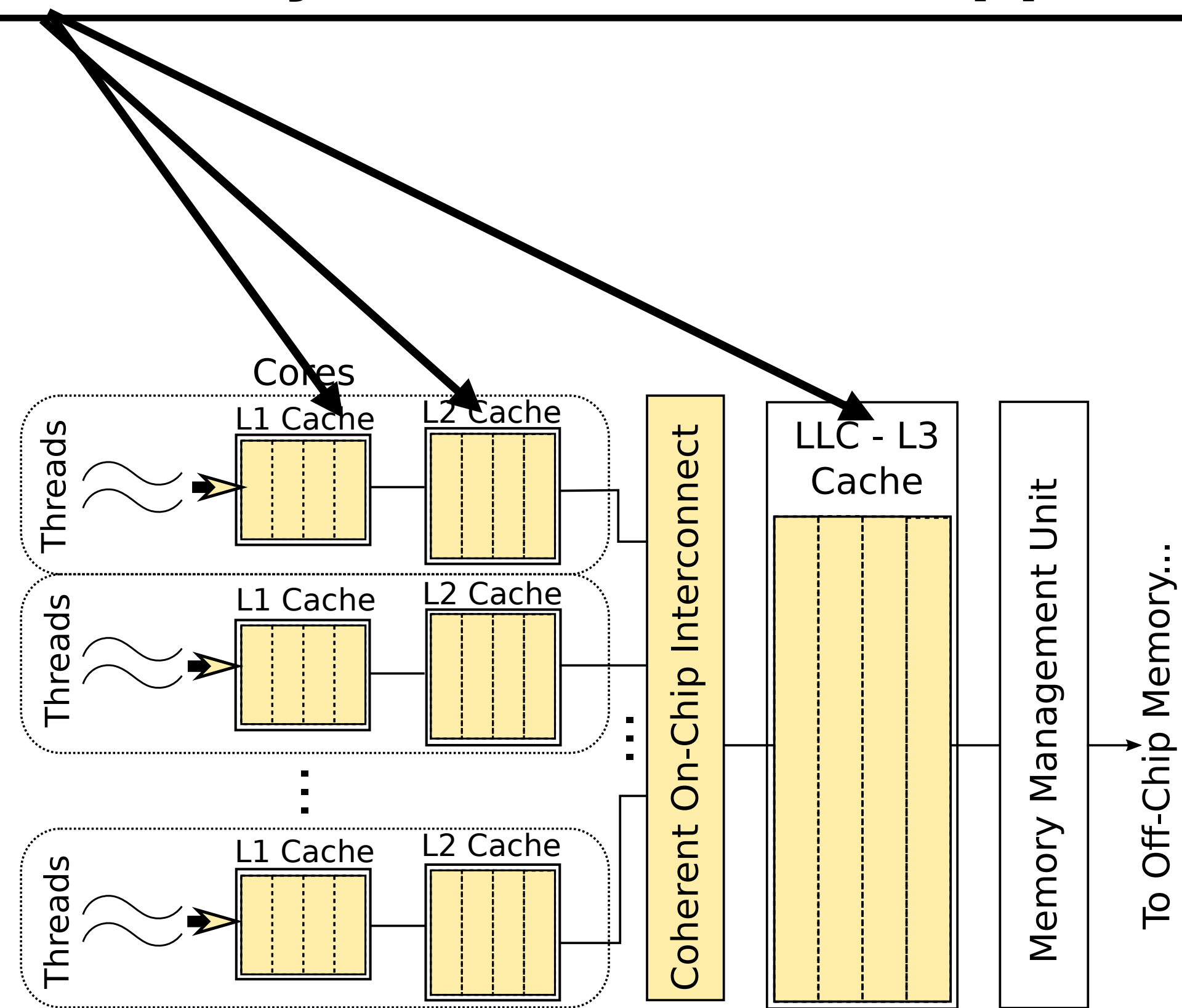
- The system software can be a part of an attack



This Talk: TEE-ShirT

How to secure all cache levels without system software support?

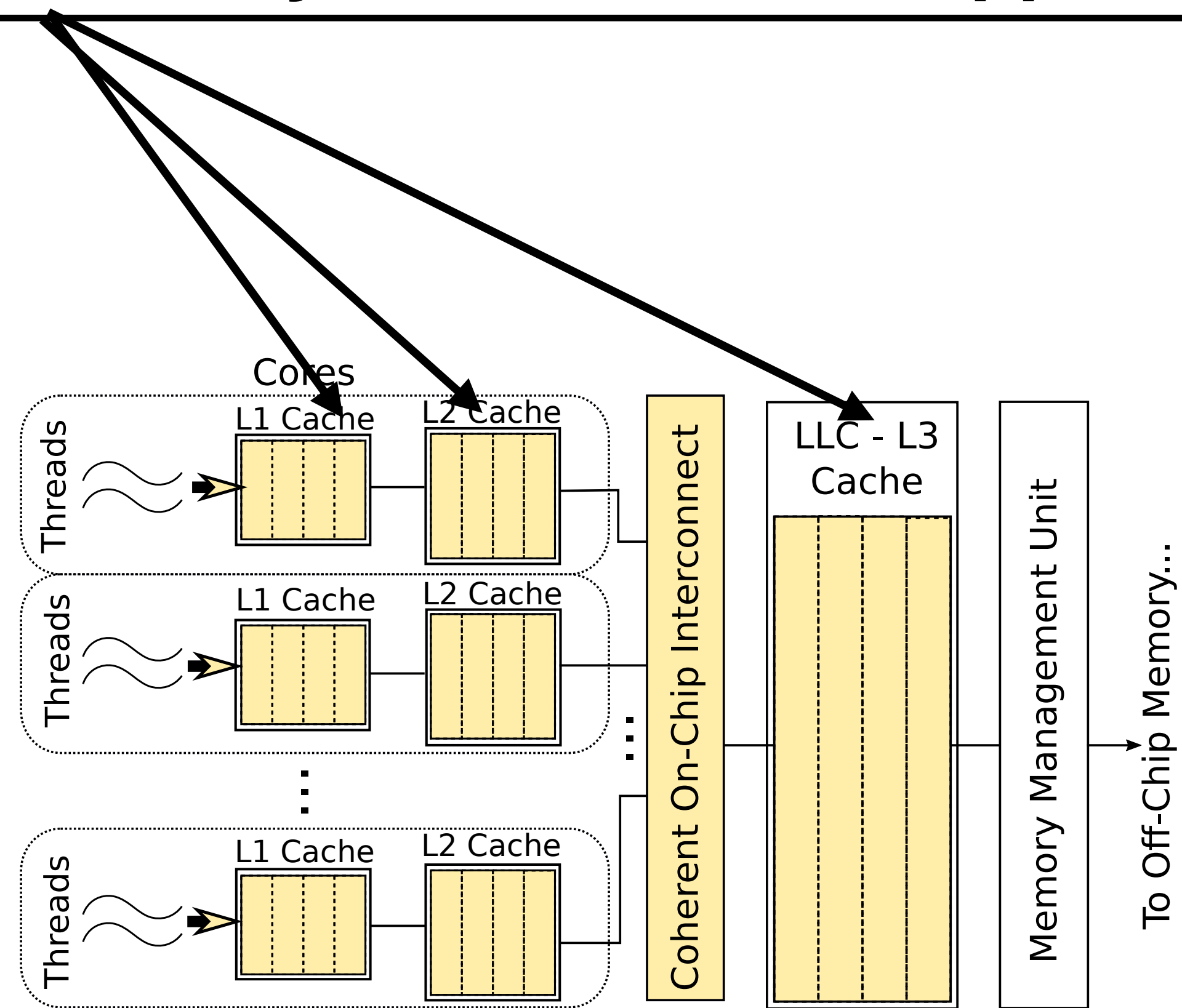
- **Scalability:** support high number of enclaves with minimal hardware
- **Context switches**
- **Cache coherence**
- **Provable security**



This Talk: TEE-ShirT

How to secure all cache levels without system software support?

- **Scalability:** support high number of enclaves with minimal hardware
- **Context switches**
- **Cache coherence**
- **Provable security**

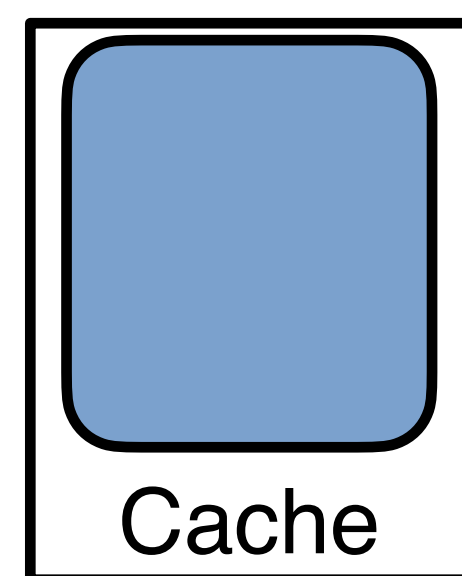
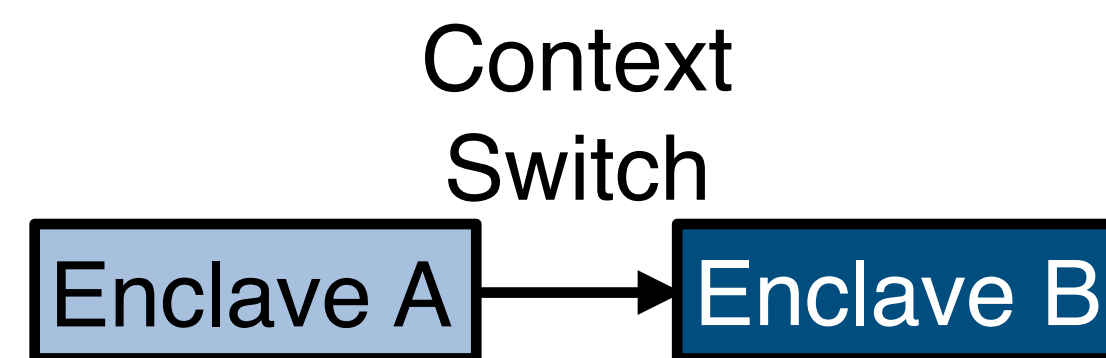


Evaluation Methodology

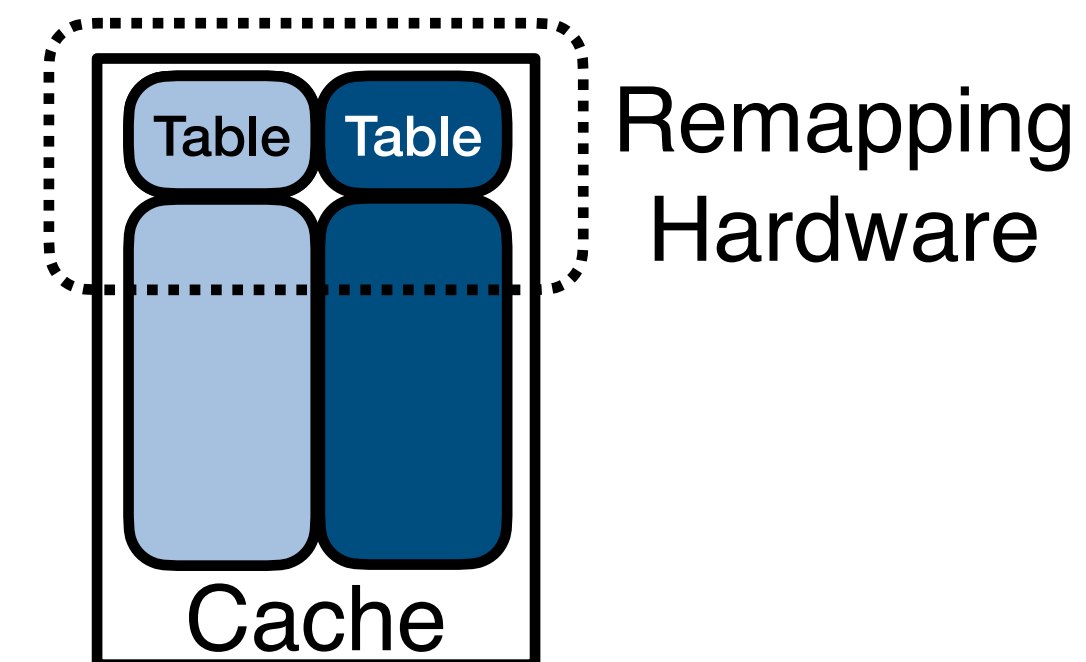
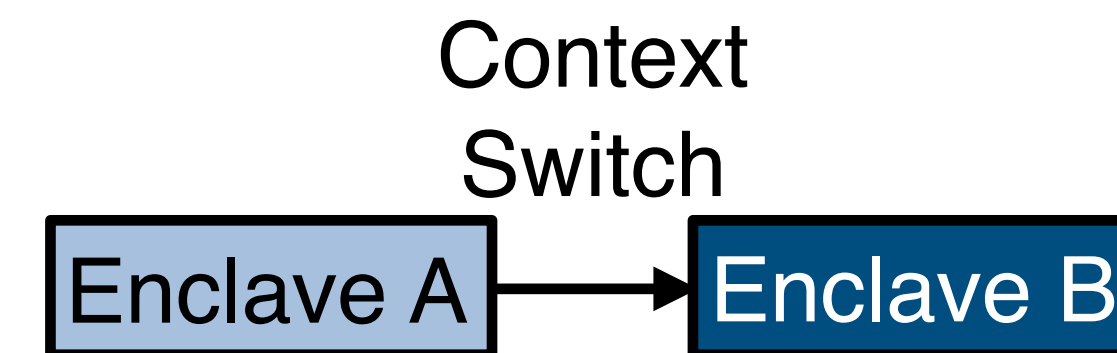
- TEE-ShirT is implemented with **gem5** cycle-accurate CPU simulator.
- We also implement TEE-ShirT's key components in an **FPGA prototype** for hardware overhead evaluation.
- We evaluate **SPEC2017** and **MiBench** benchmark suites.
- We mechanize operational semantics for the formal security proof with **Coq**.

Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



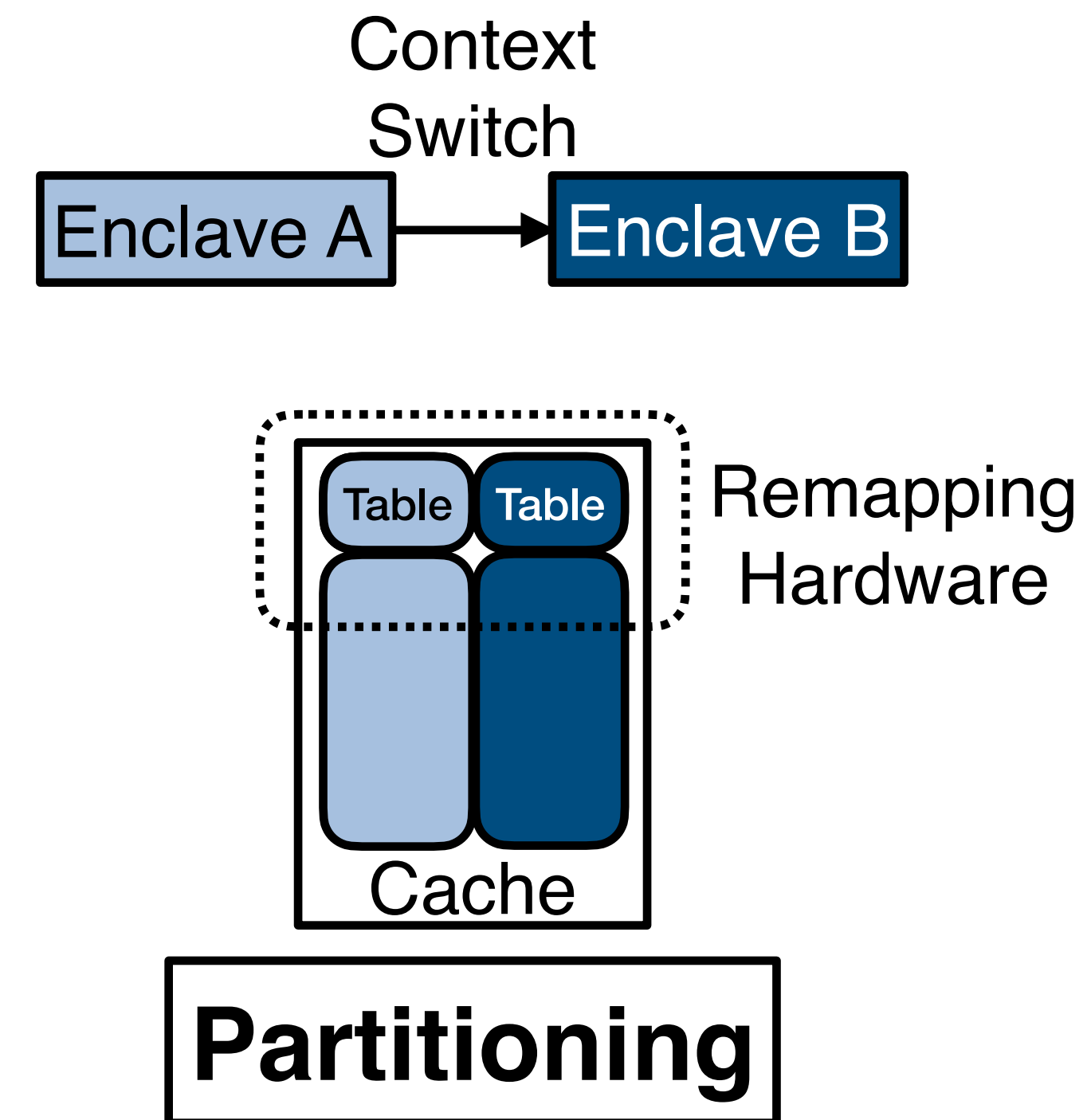
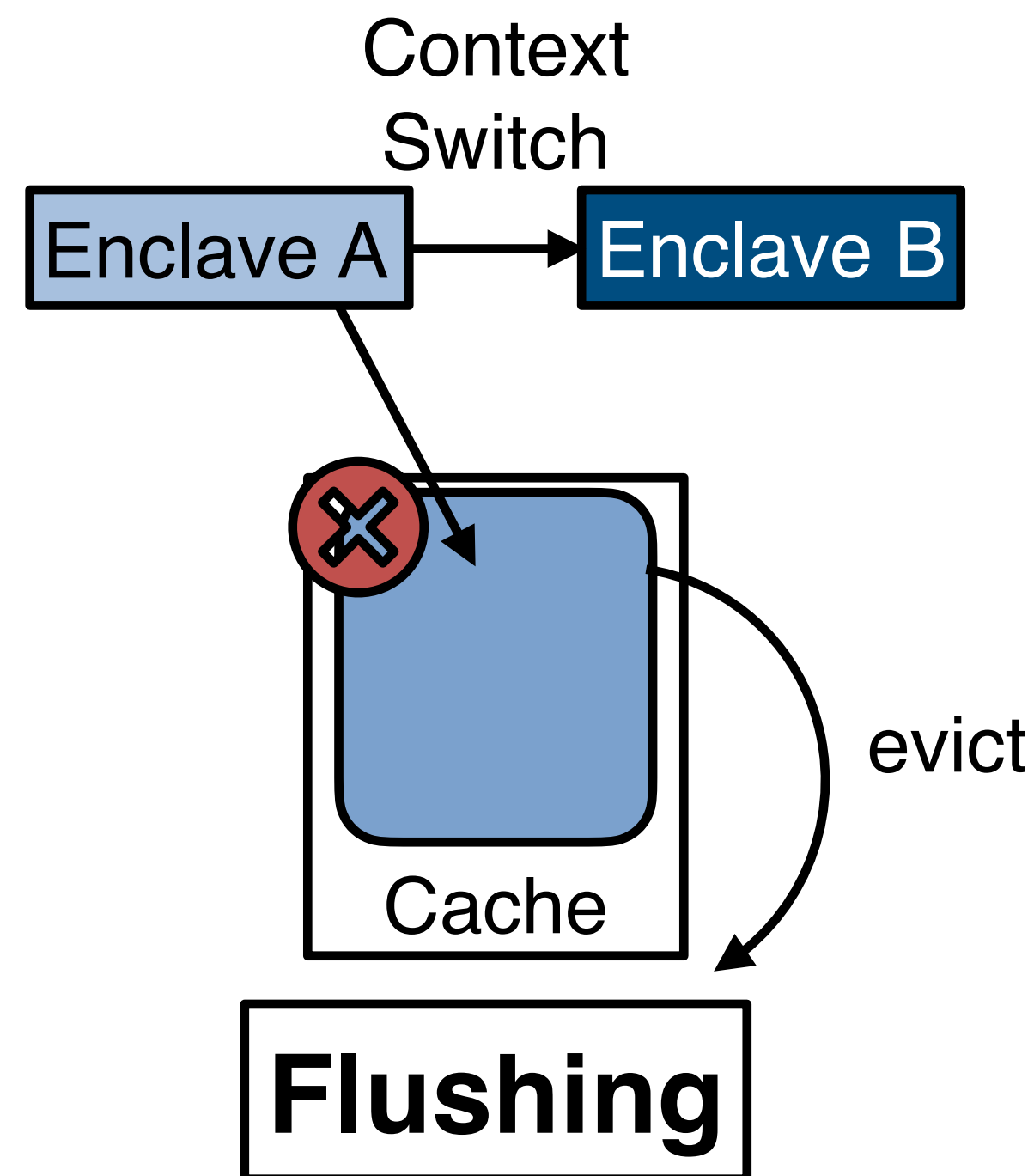
Flushing



Partitioning

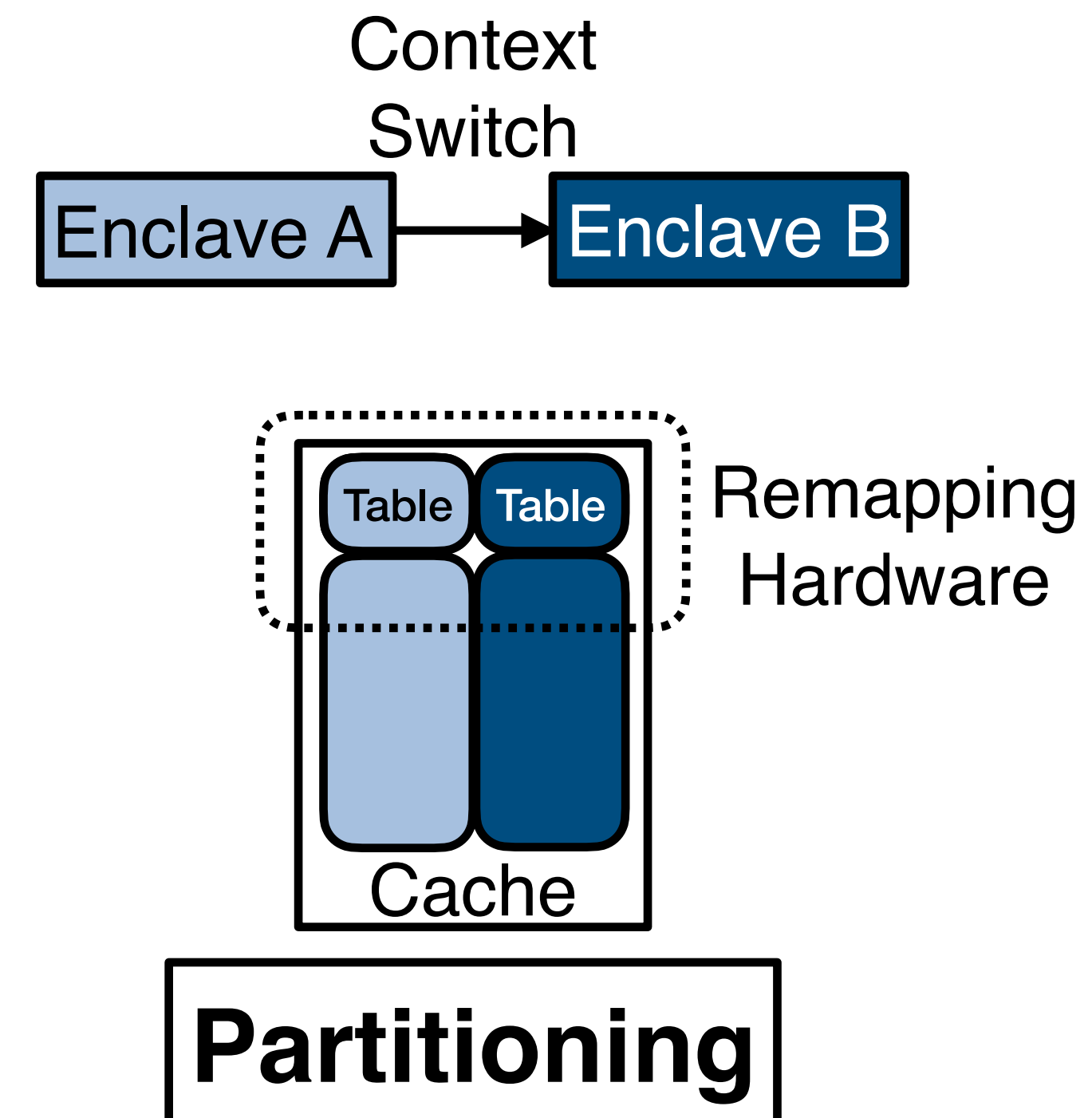
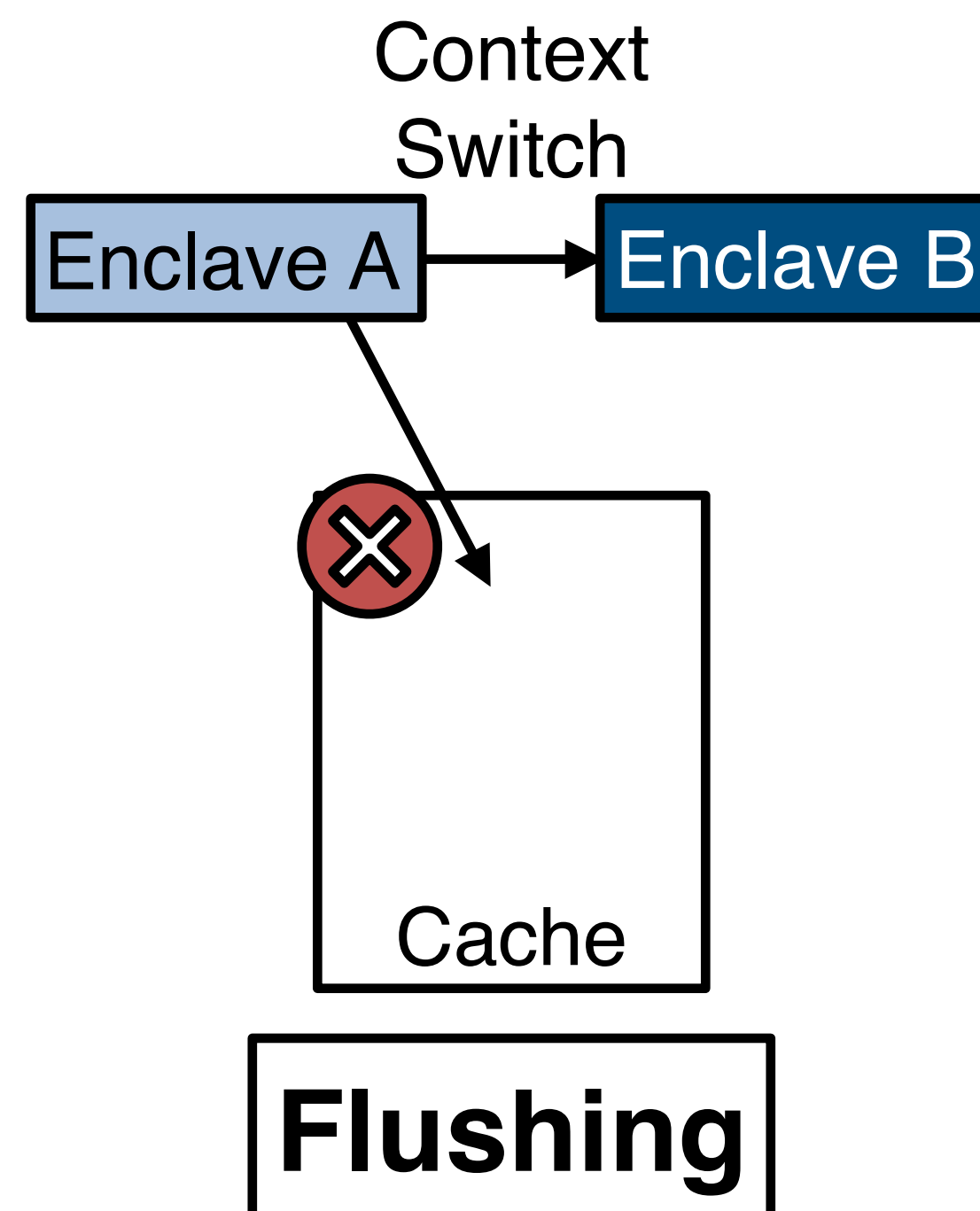
Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



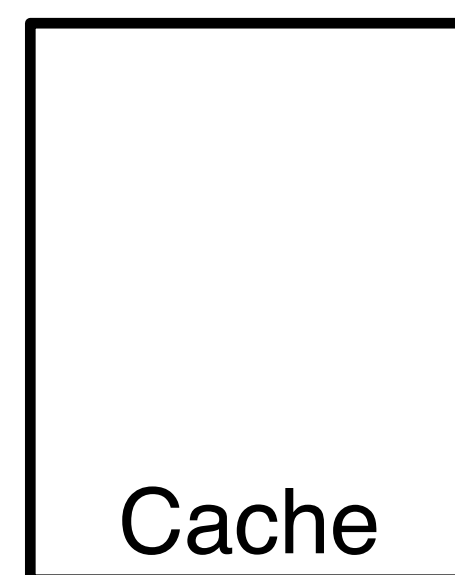
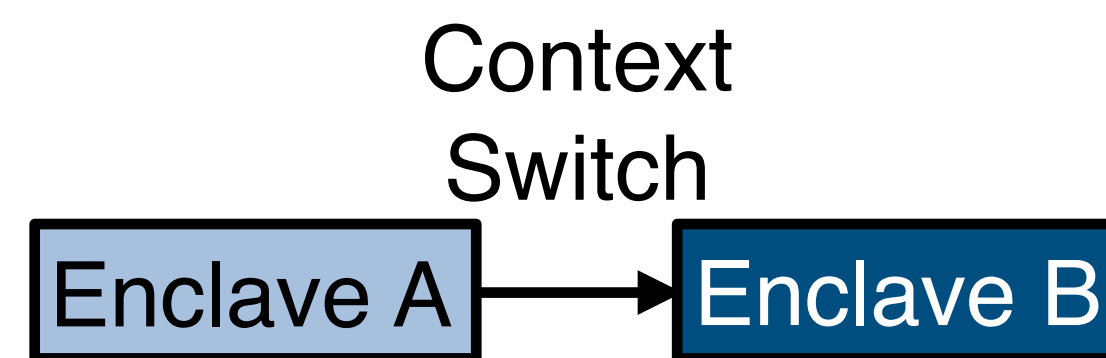
Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.

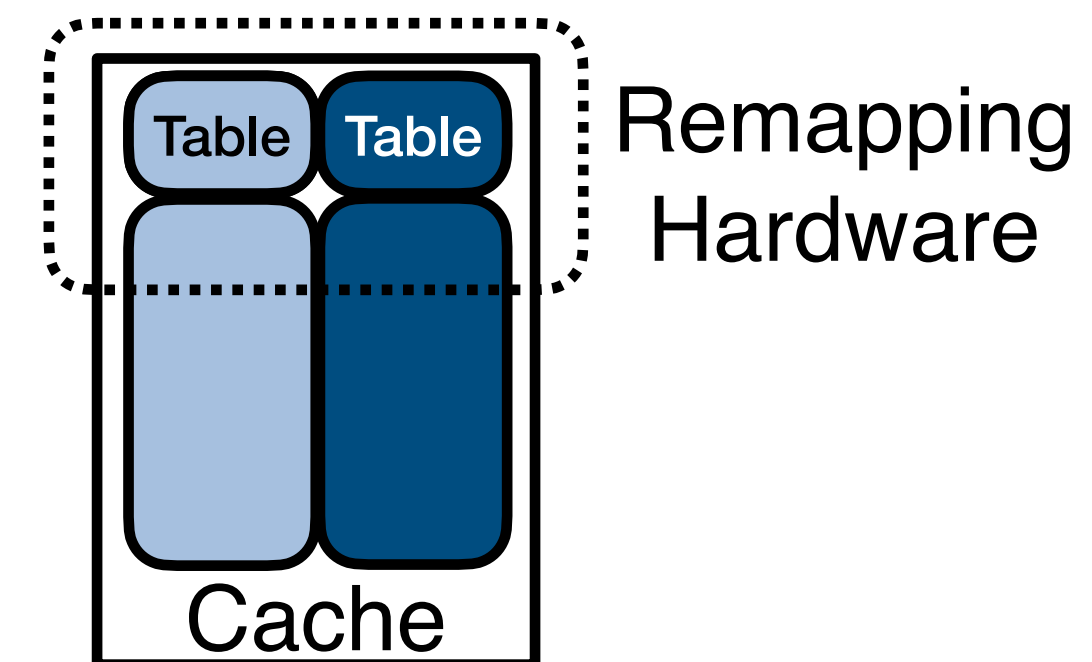
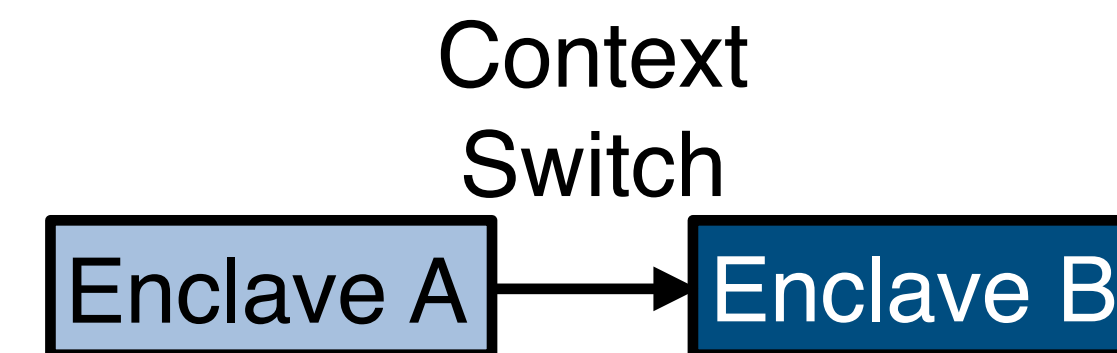


Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



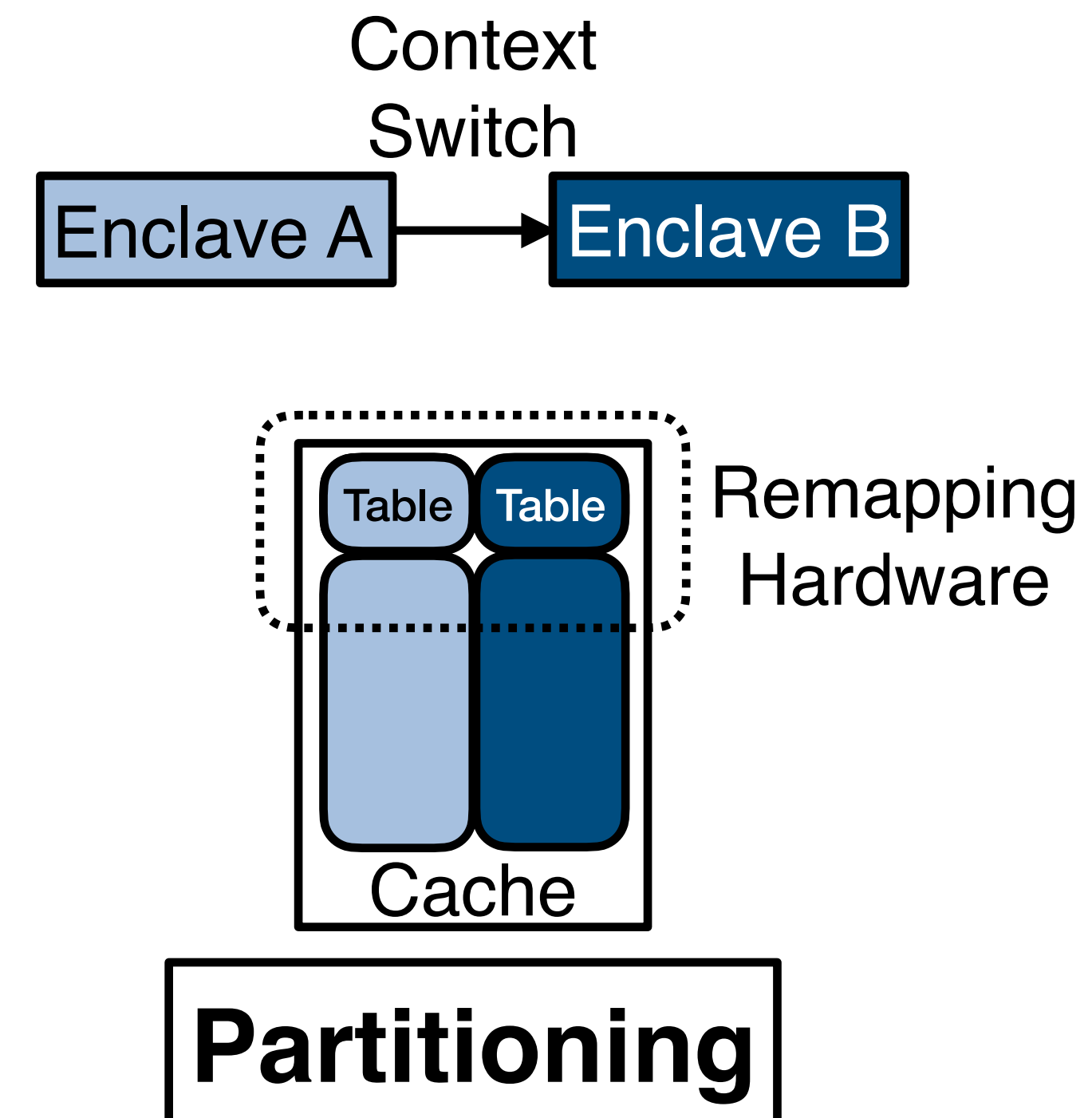
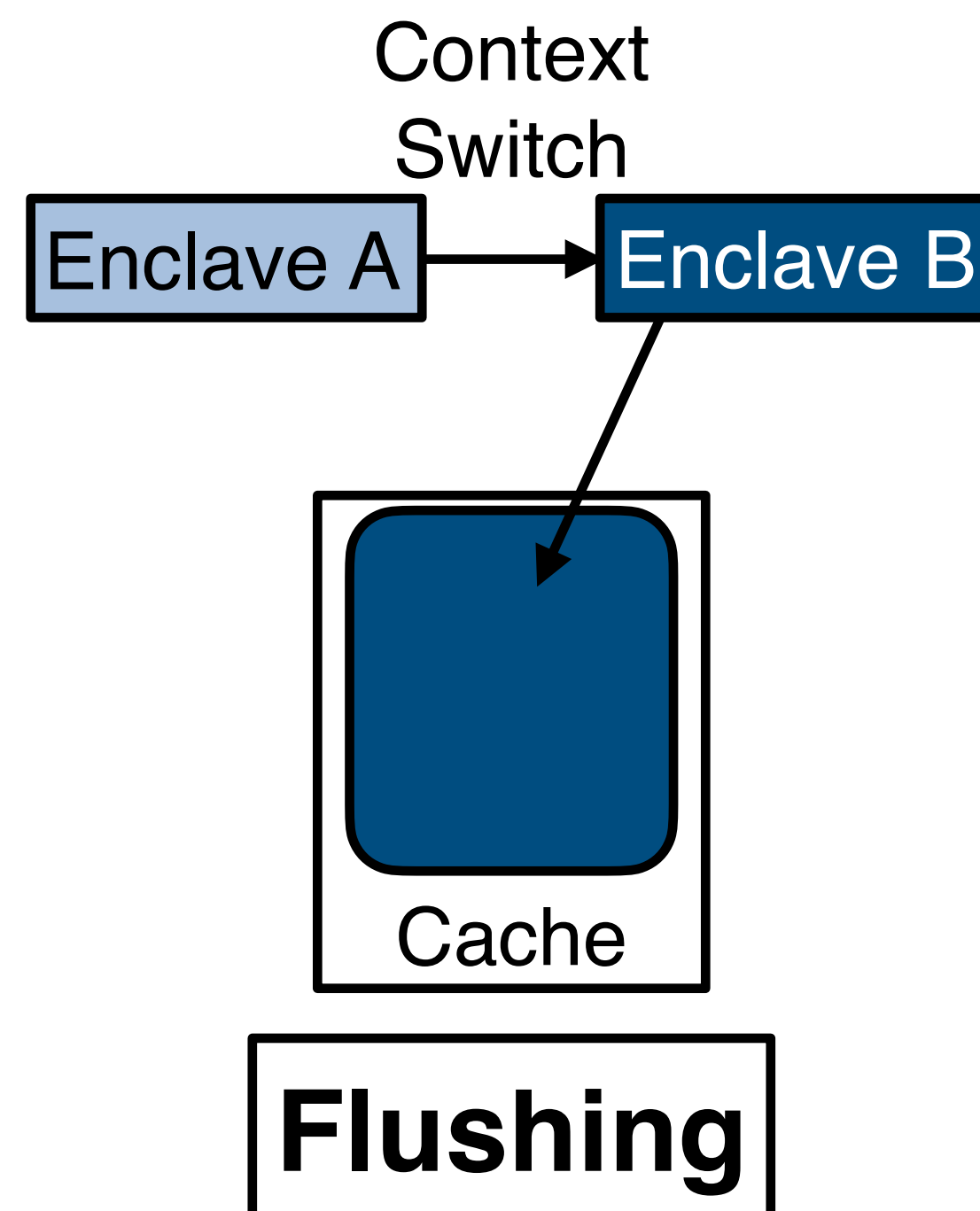
Flushing



Partitioning

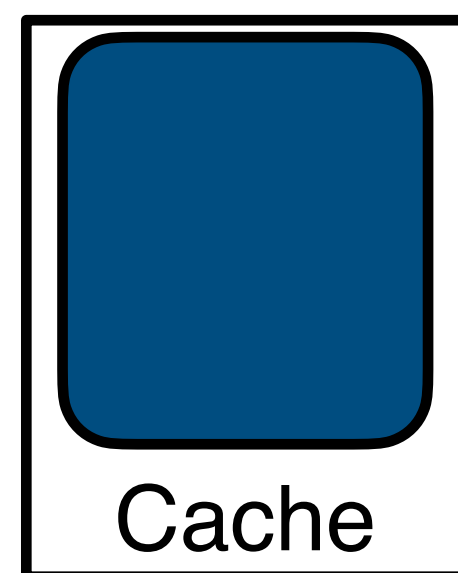
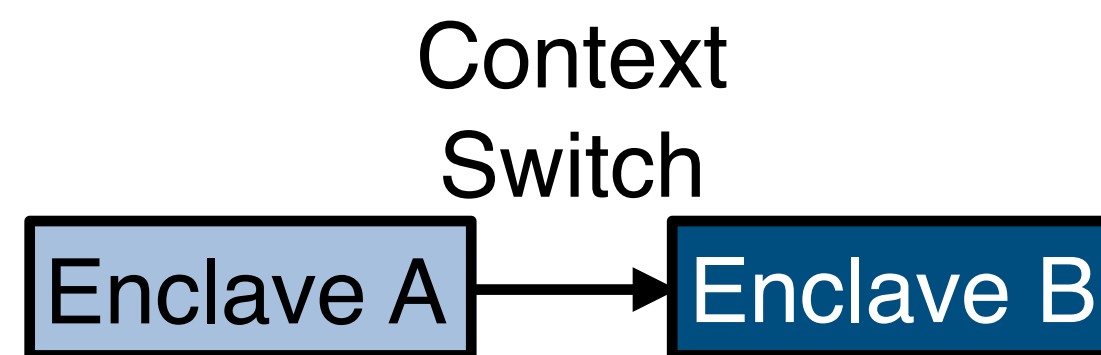
Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.

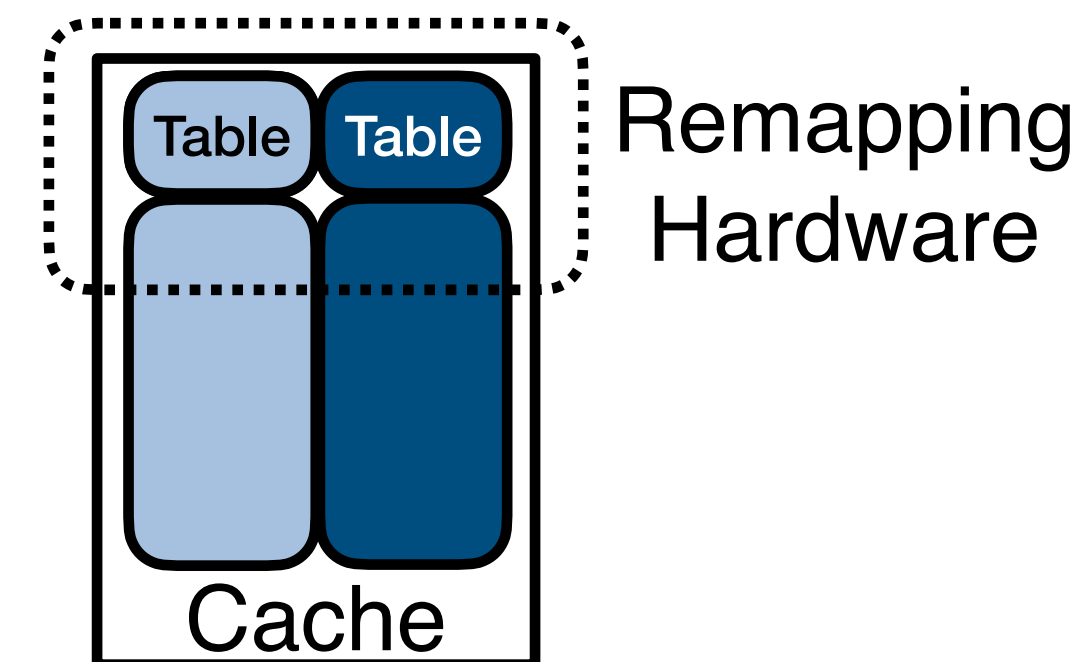
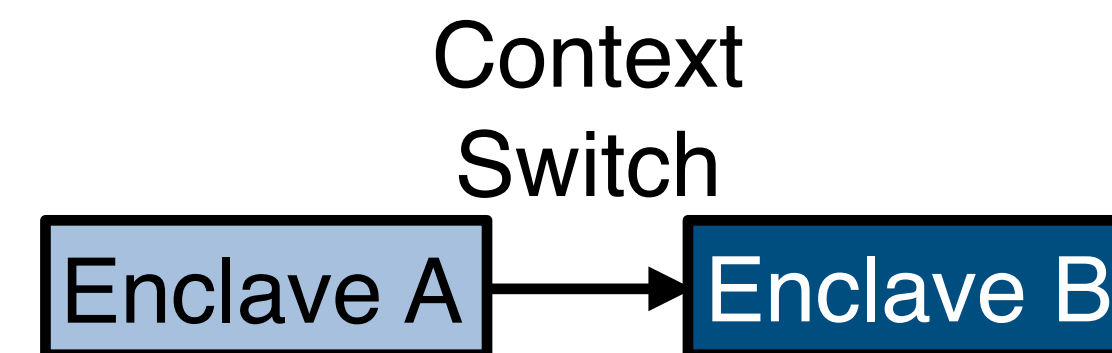


Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



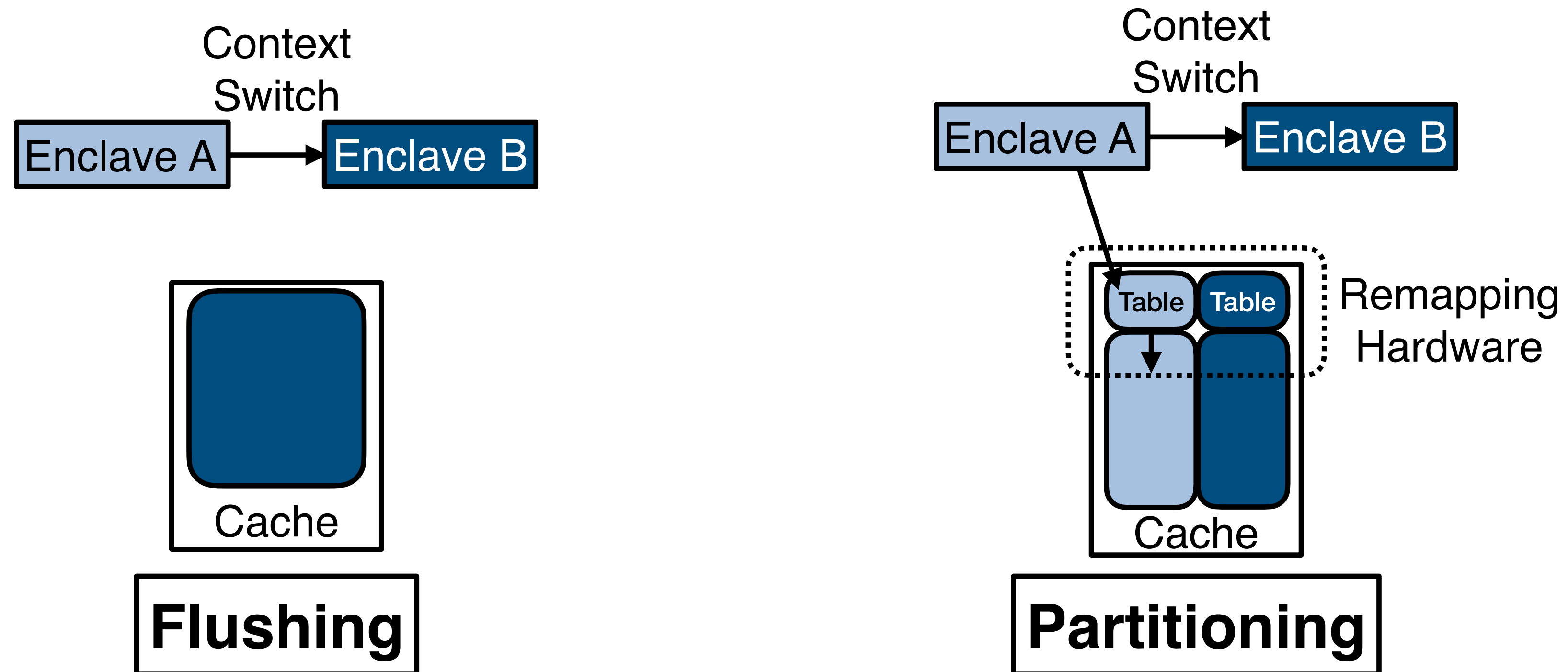
Flushing



Partitioning

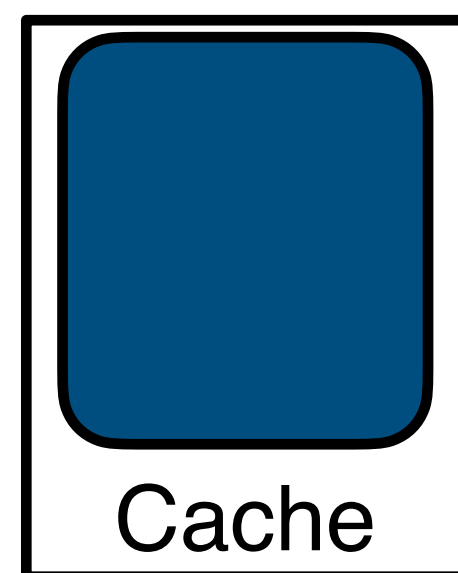
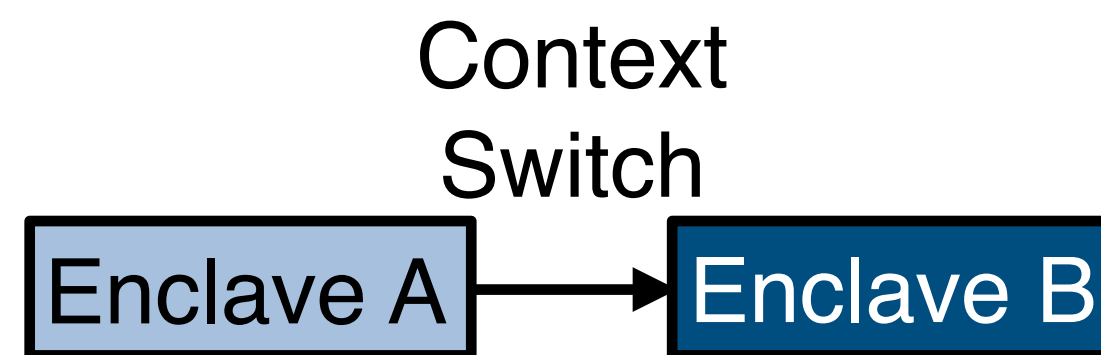
Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.

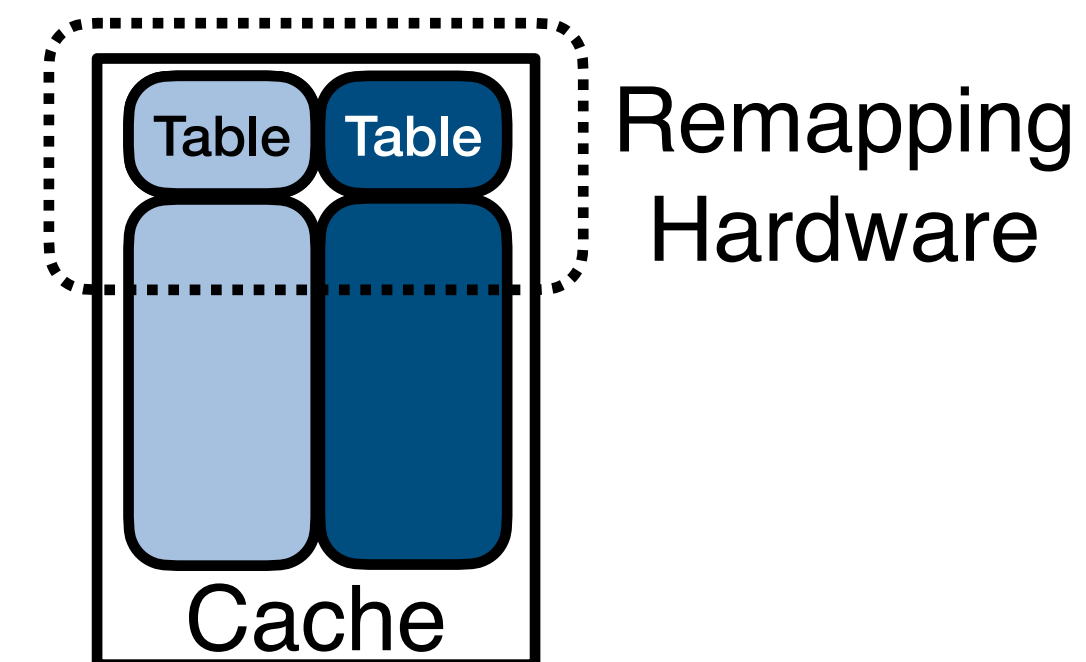
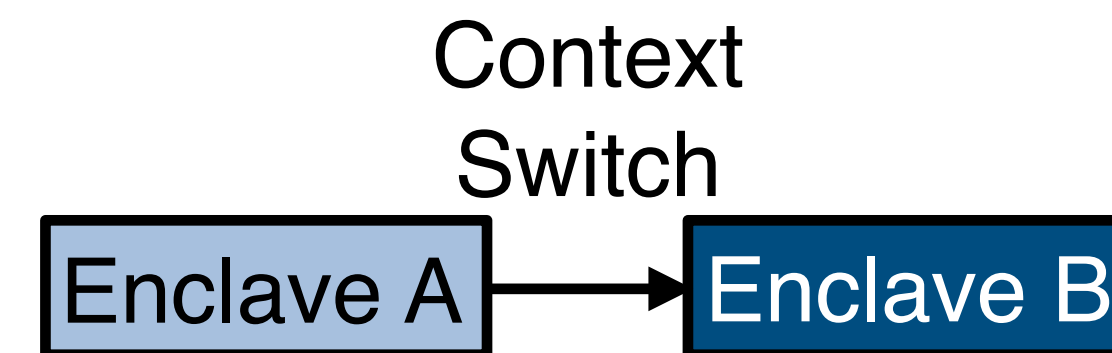


Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



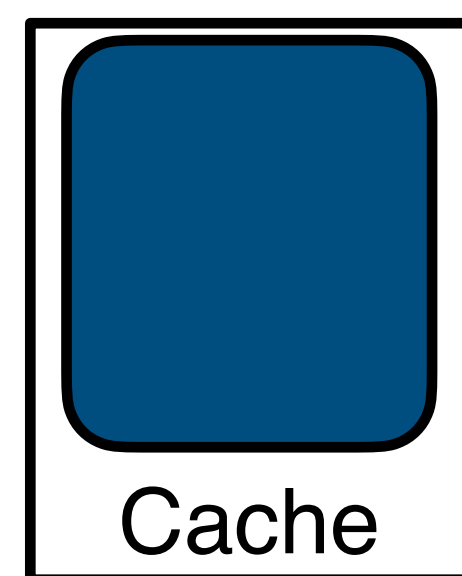
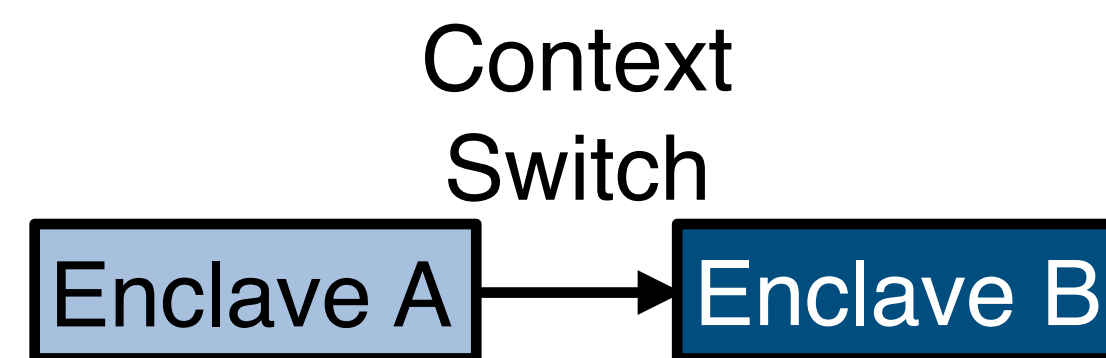
Flushing



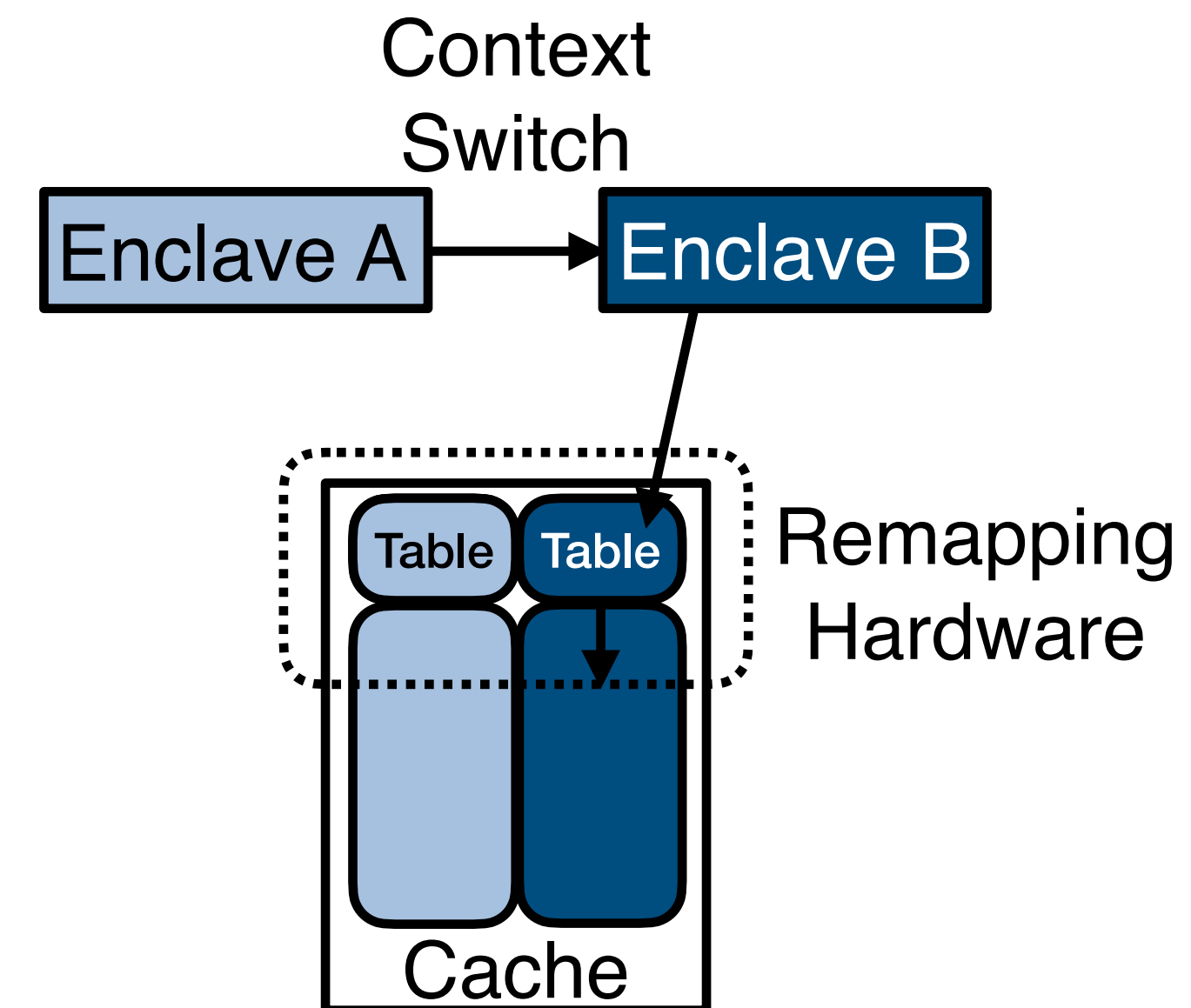
Partitioning

Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



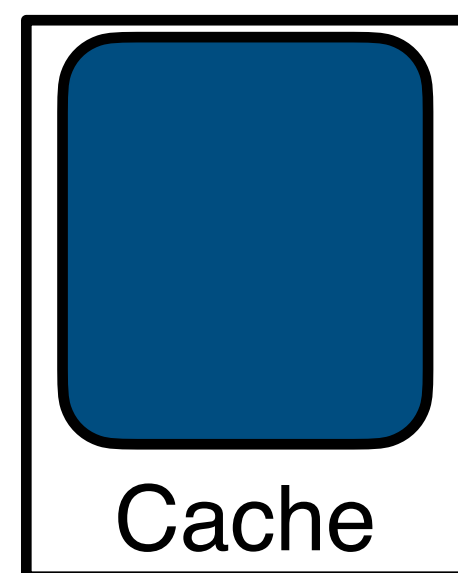
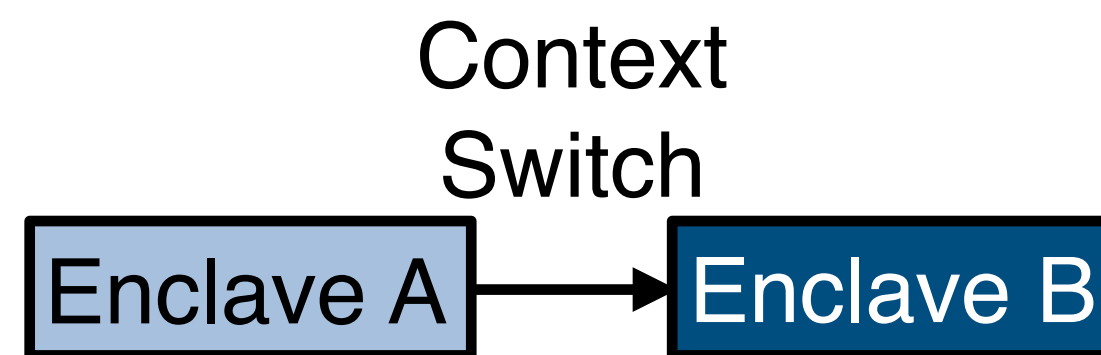
Flushing



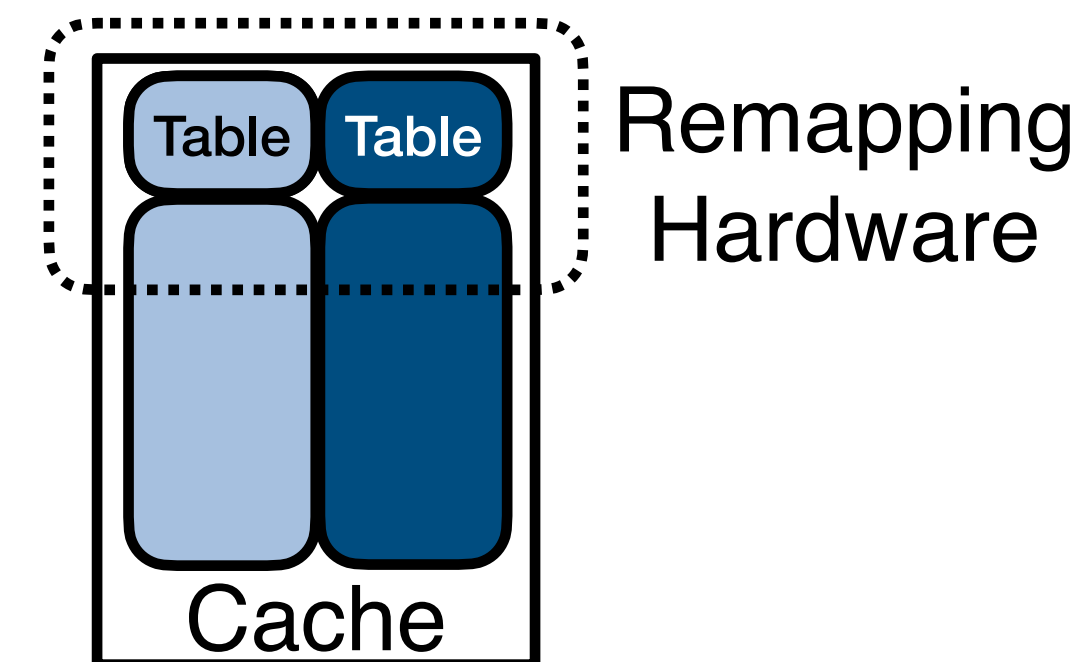
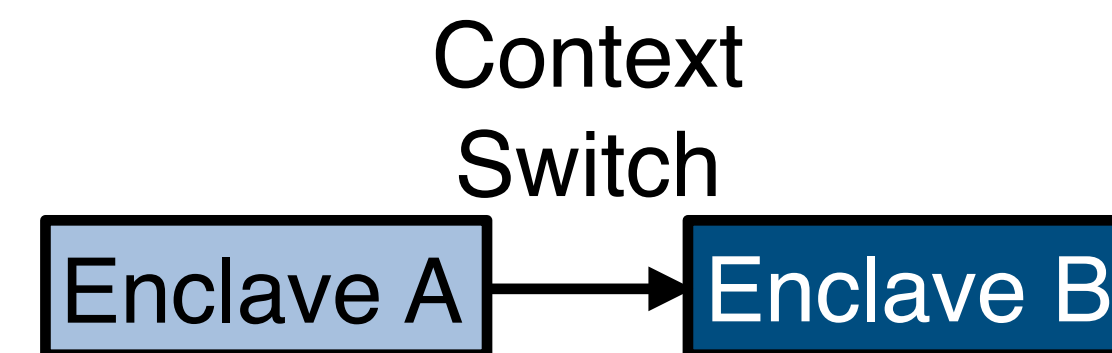
Partitioning

Two Approaches to Secure Caches

- **Flushing:** invalidate and write-back dirty data.
- **Partitioning:** divide security domains into dedicated cache boundaries.



Flushing



Partitioning

How to Secure Private Caches?

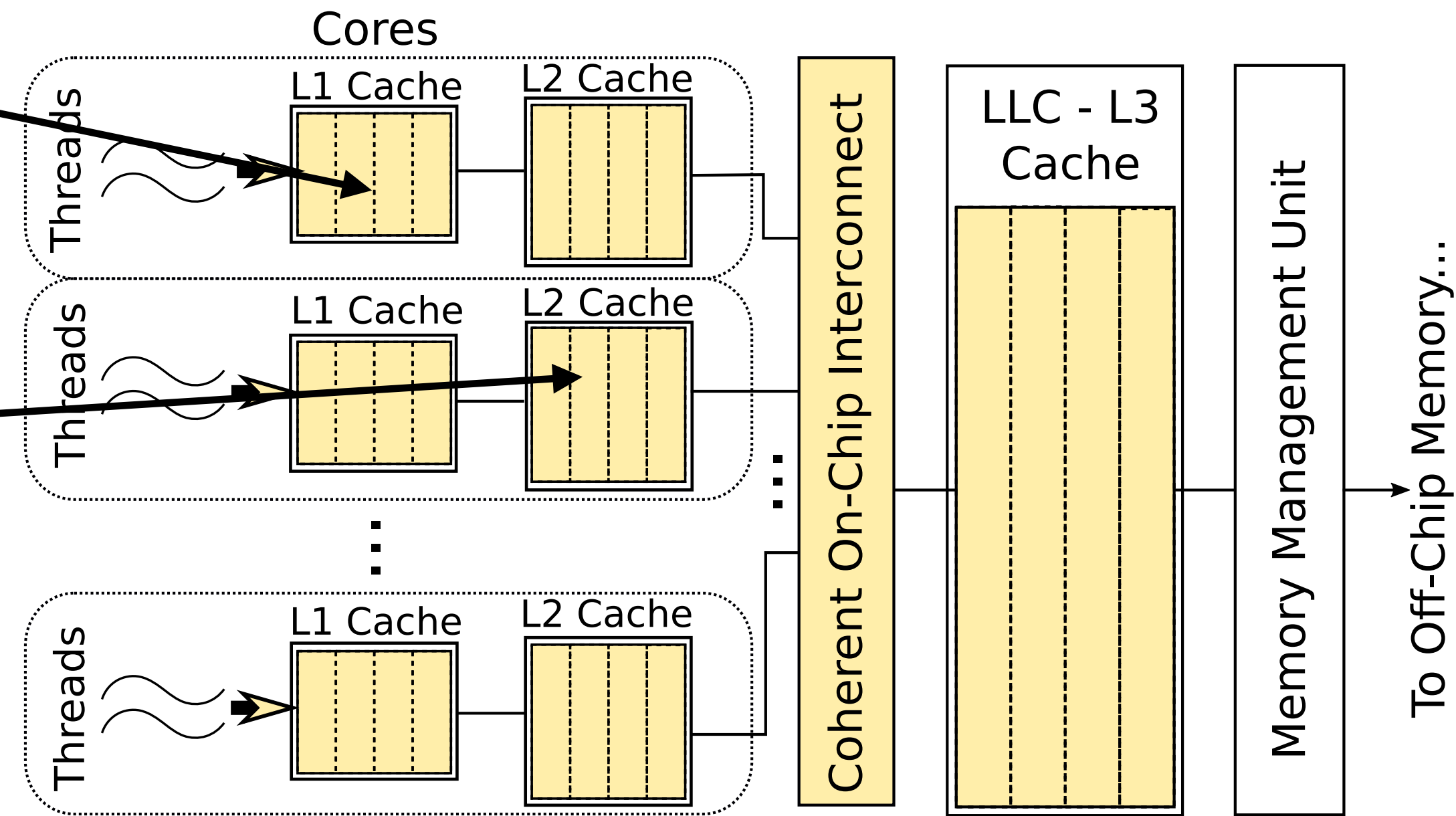
- There are two levels of private caches: **L1** and **L2**

- **L1:**

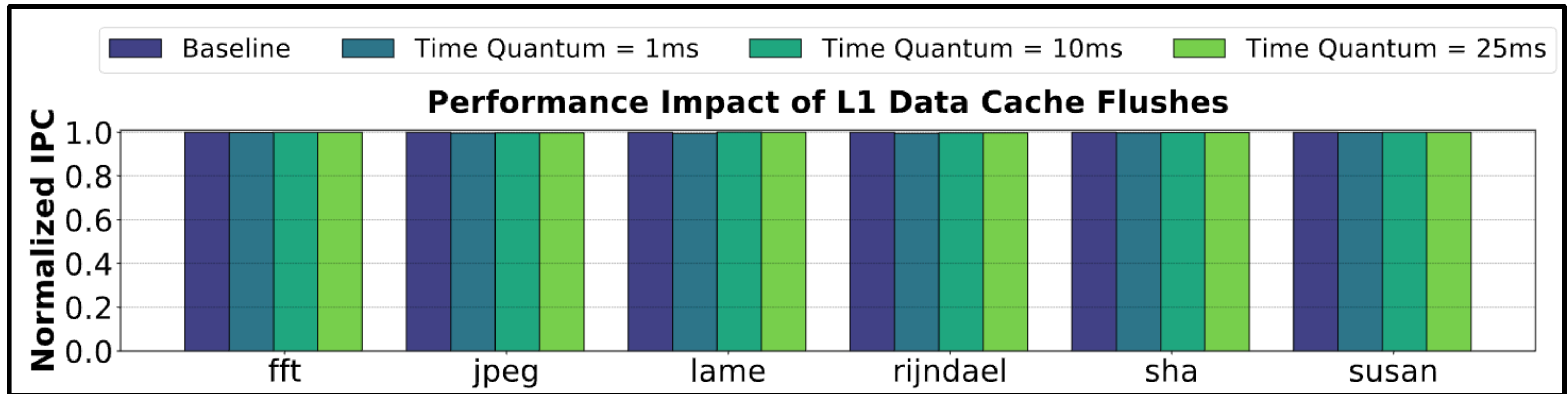
- Frequent accesses
- Low latency access
- Typically 2-64KB

- **L2:**

- High latency access
- Typically 256-512KB

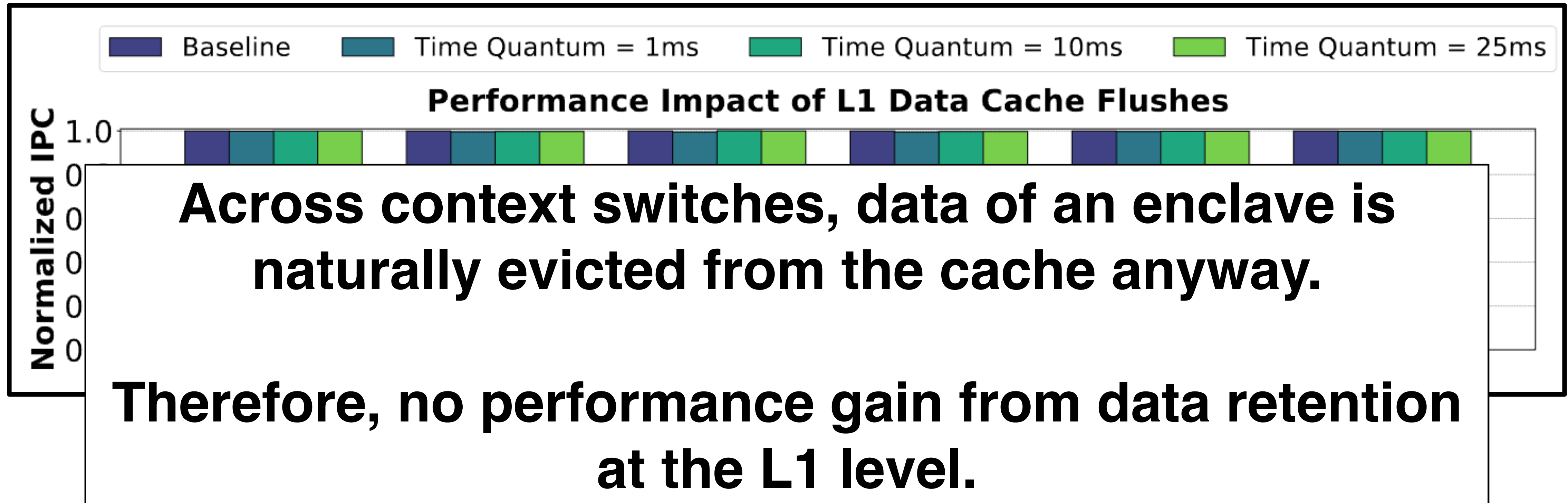


TEE-SHirT L1 Caches: Flushing

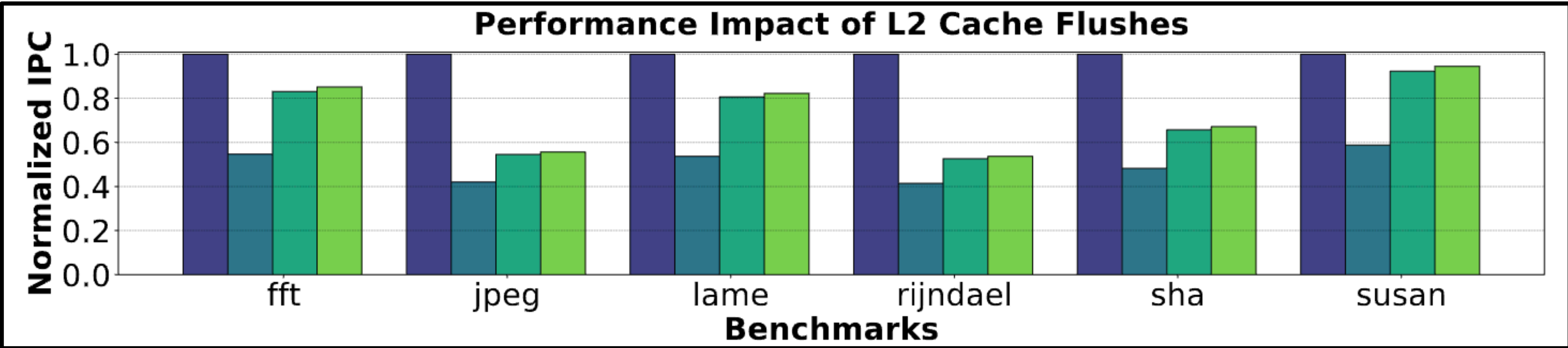


Flushing across context switches and system calls is a sufficient solution for the L1 cache.

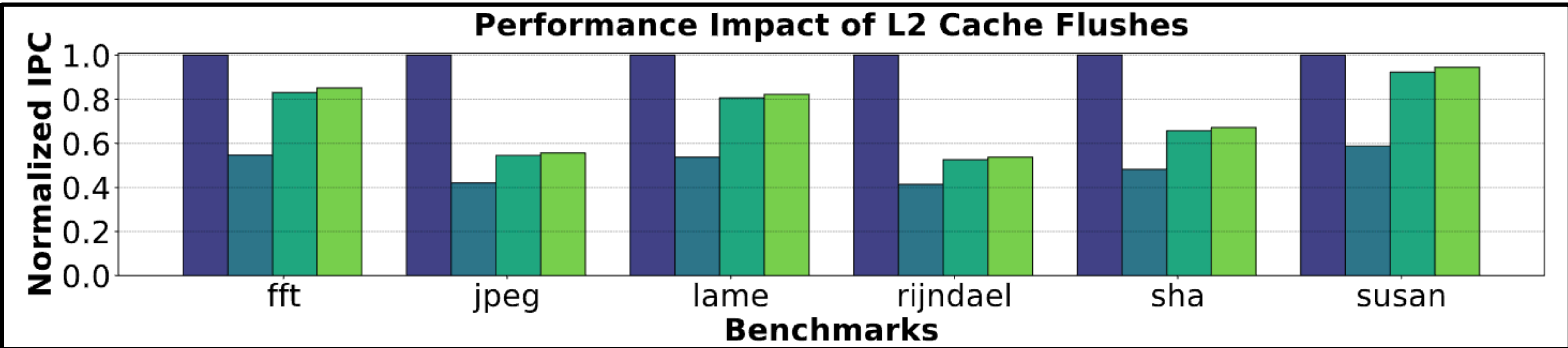
TEE-SHirT L1 Caches: Flushing



Can We Do the Same to L2 Caches?



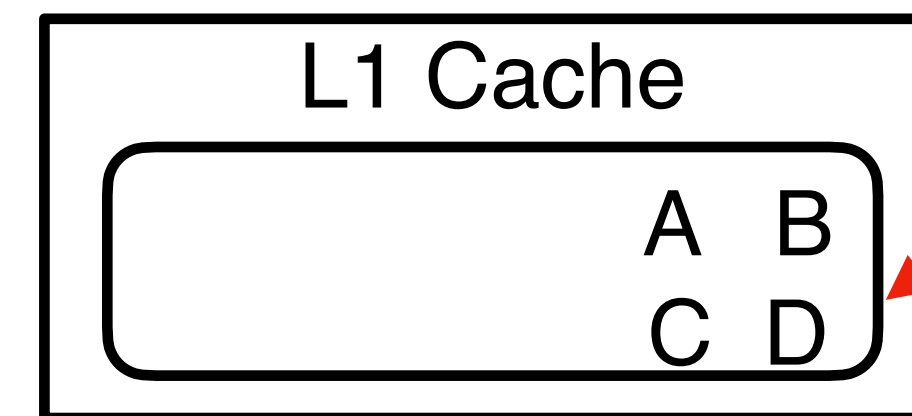
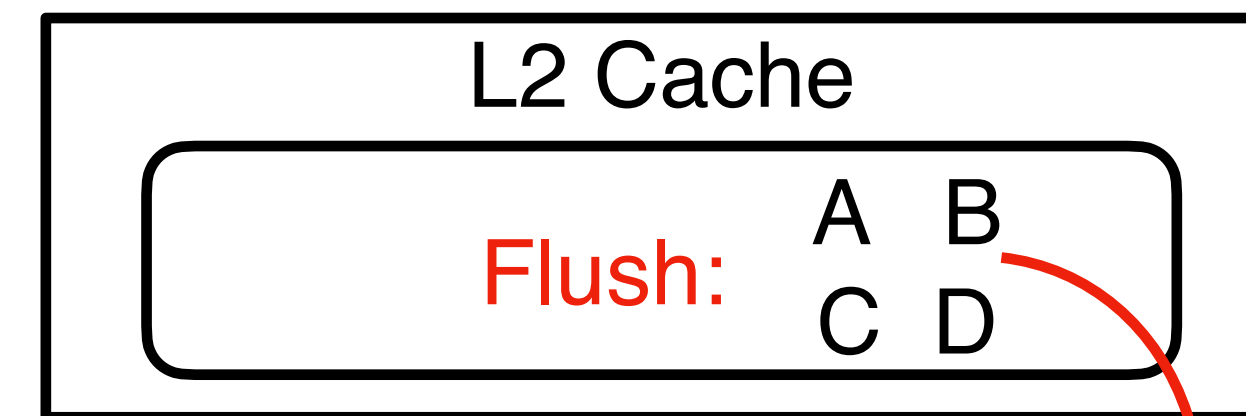
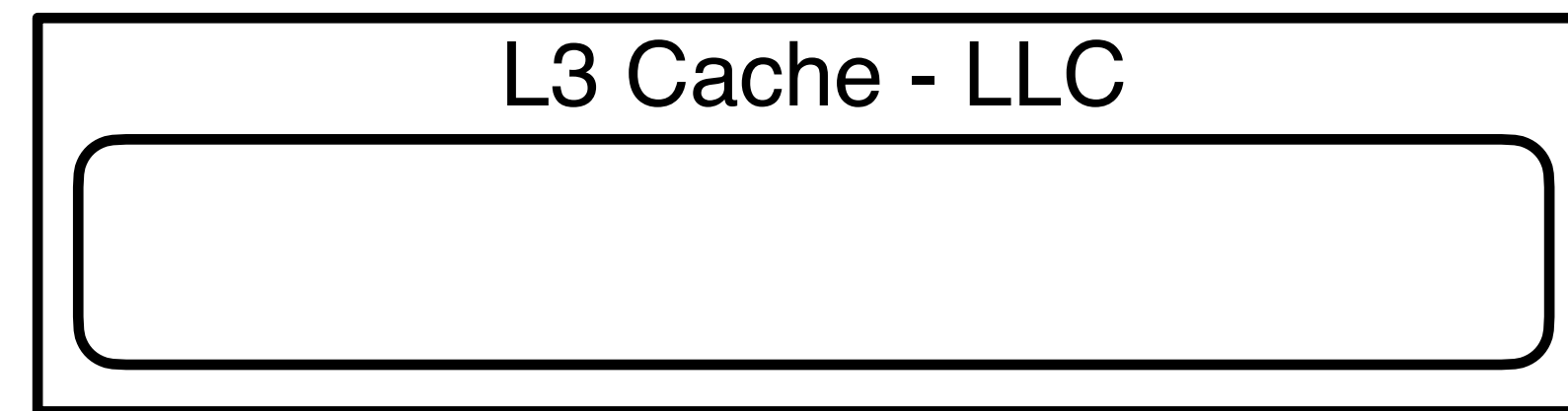
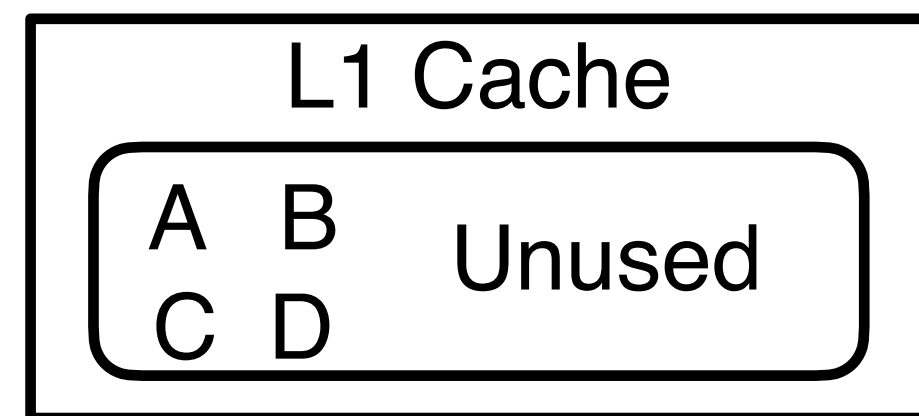
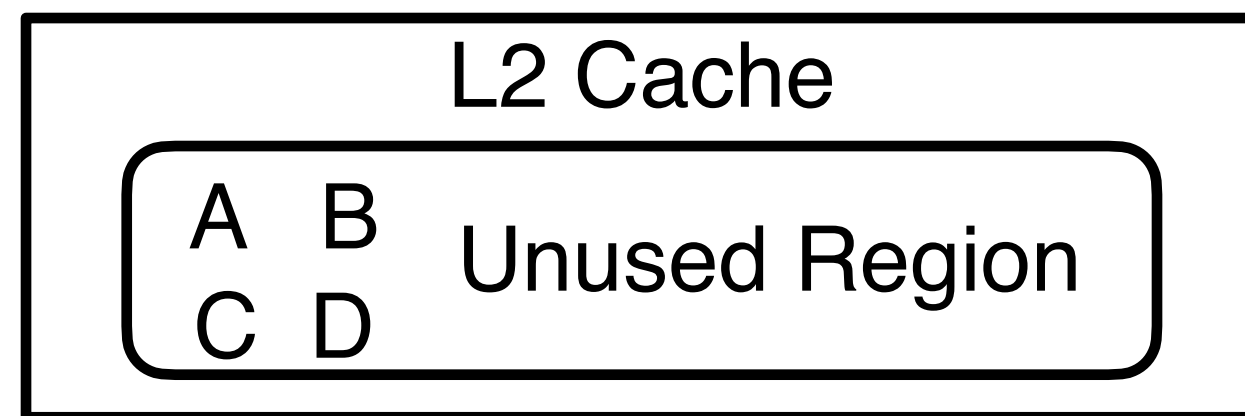
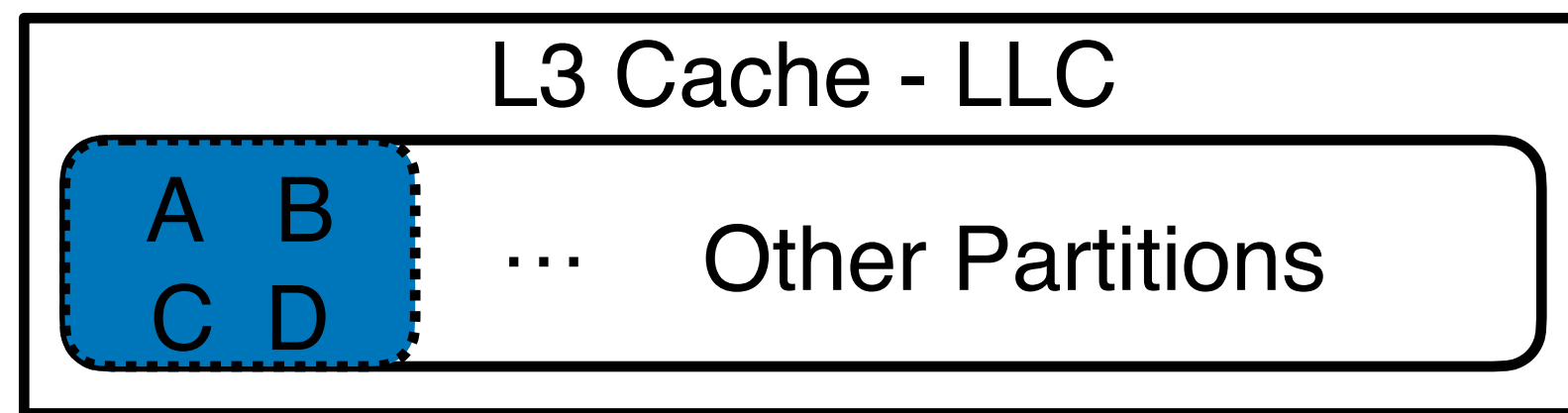
Can We Do the Same to L2 Caches?



Why is it so expensive?

TEE-ShirT L2 Caches: Partitioning

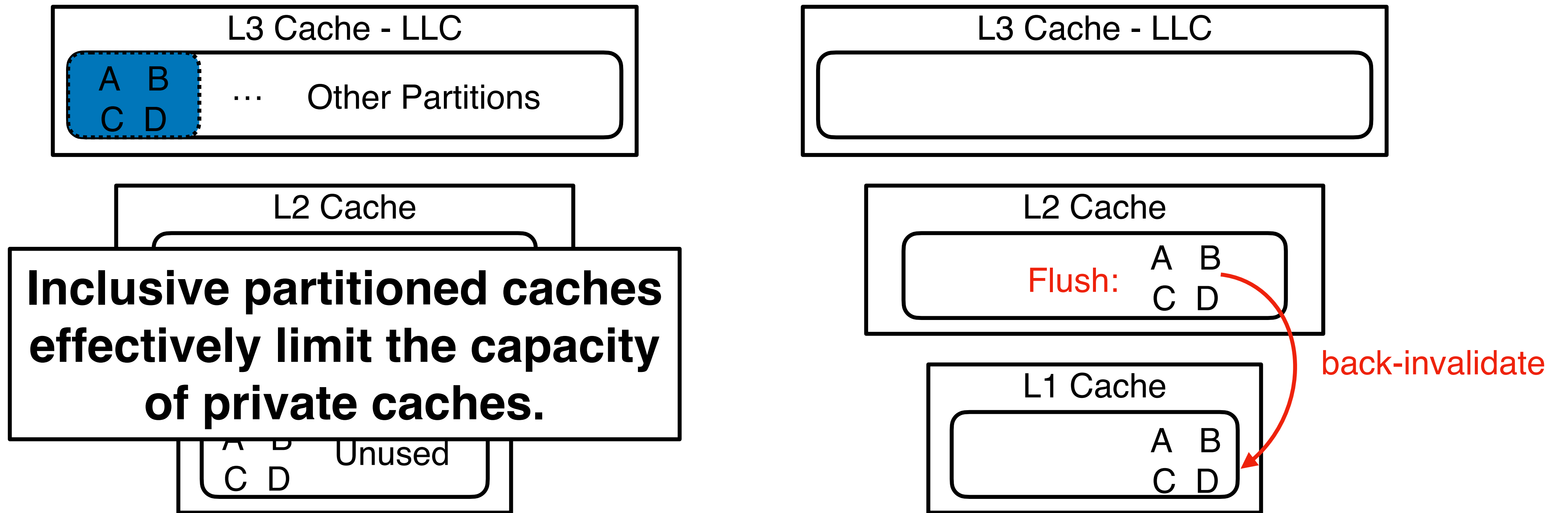
- **Inclusivity Property:** Lines in a lower level structure should also be present in the higher levels.



back-invalidate

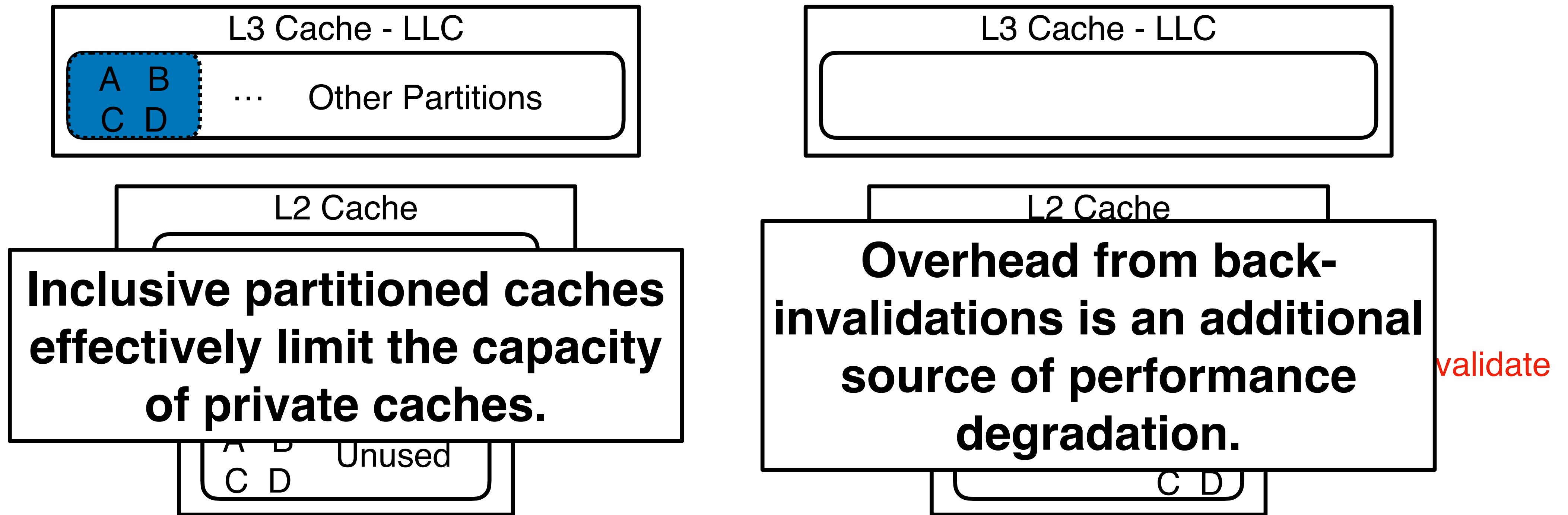
TEE-ShirT L2 Caches: Partitioning

- **Inclusivity Property:** Lines in a lower level structure should also be present in the higher levels.



TEE-ShirT L2 Caches: Partitioning

- **Inclusivity Property:** Lines in a lower level structure should also be present in the higher levels.



TEE-SHirT LLC: Fine-Grain Partitioning

- Examples:
- **Bespoke Cache Enclaves** (SEED'21)
- **Chunked-Cache** (NDSS'22)
- **Composable Cachelets** (USENIX Sec'22)

TEE-SHirT LLC: Fine-Grain Partitioning

- Examples:
- **Bespoke Cache Enclaves** (SEED'21)
- **Chunked-Cache** (NDSS'22)
- **Composable Cachelets** (USENIX Sec'22)

Prior literature already establishes that LLC has to be partitioned.

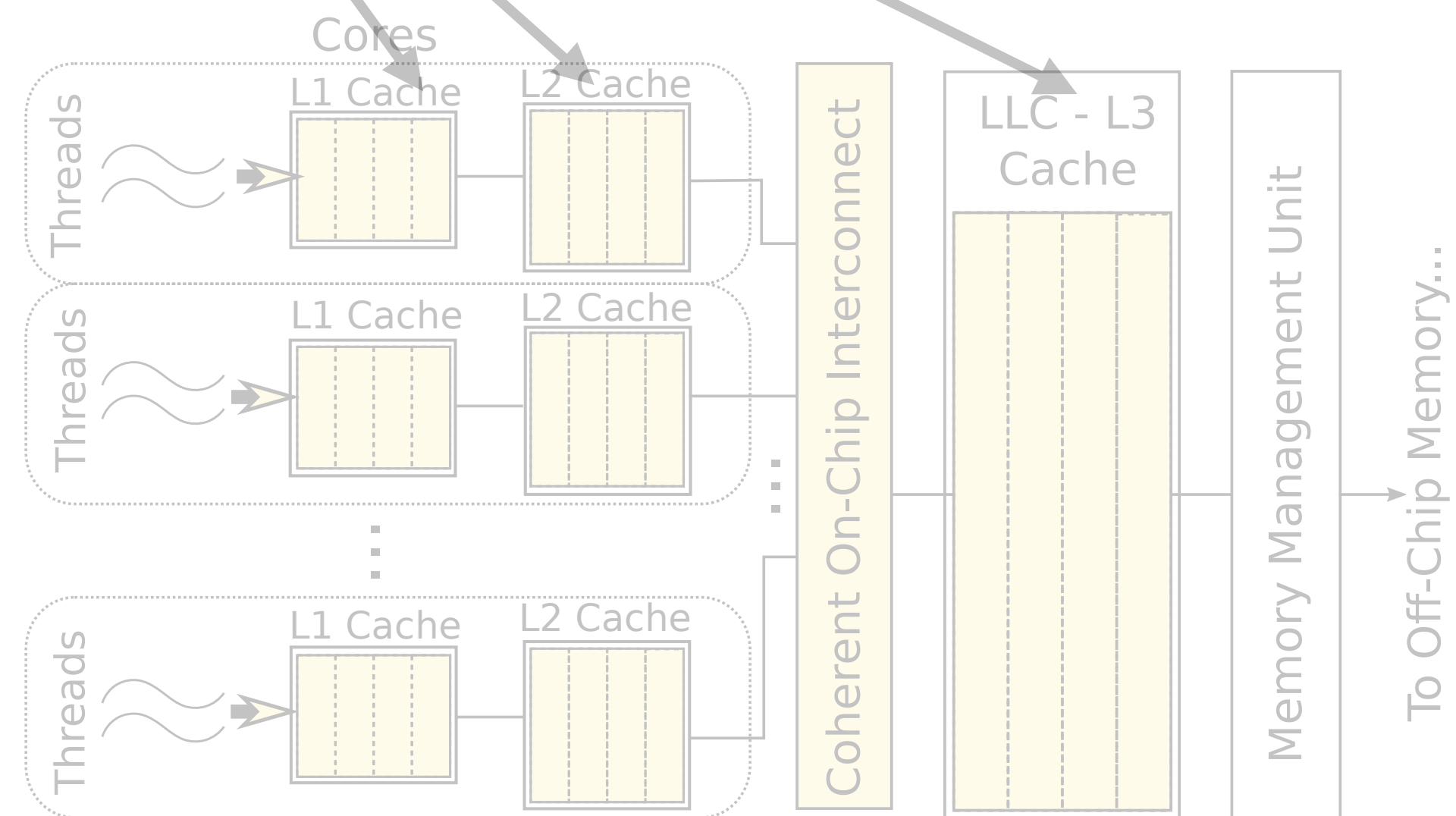
TEE-ShirT Cache Hierarchy

- **L1 Data/Instruction Cache:** Flush across context switches.
- **L2 Cache:** Cache partitioning
- **L3 Cache (LLC):** Cache Partitioning
- We also ensure **cache coherence** across levels (details in the paper)

This Talk: TEE-ShirT

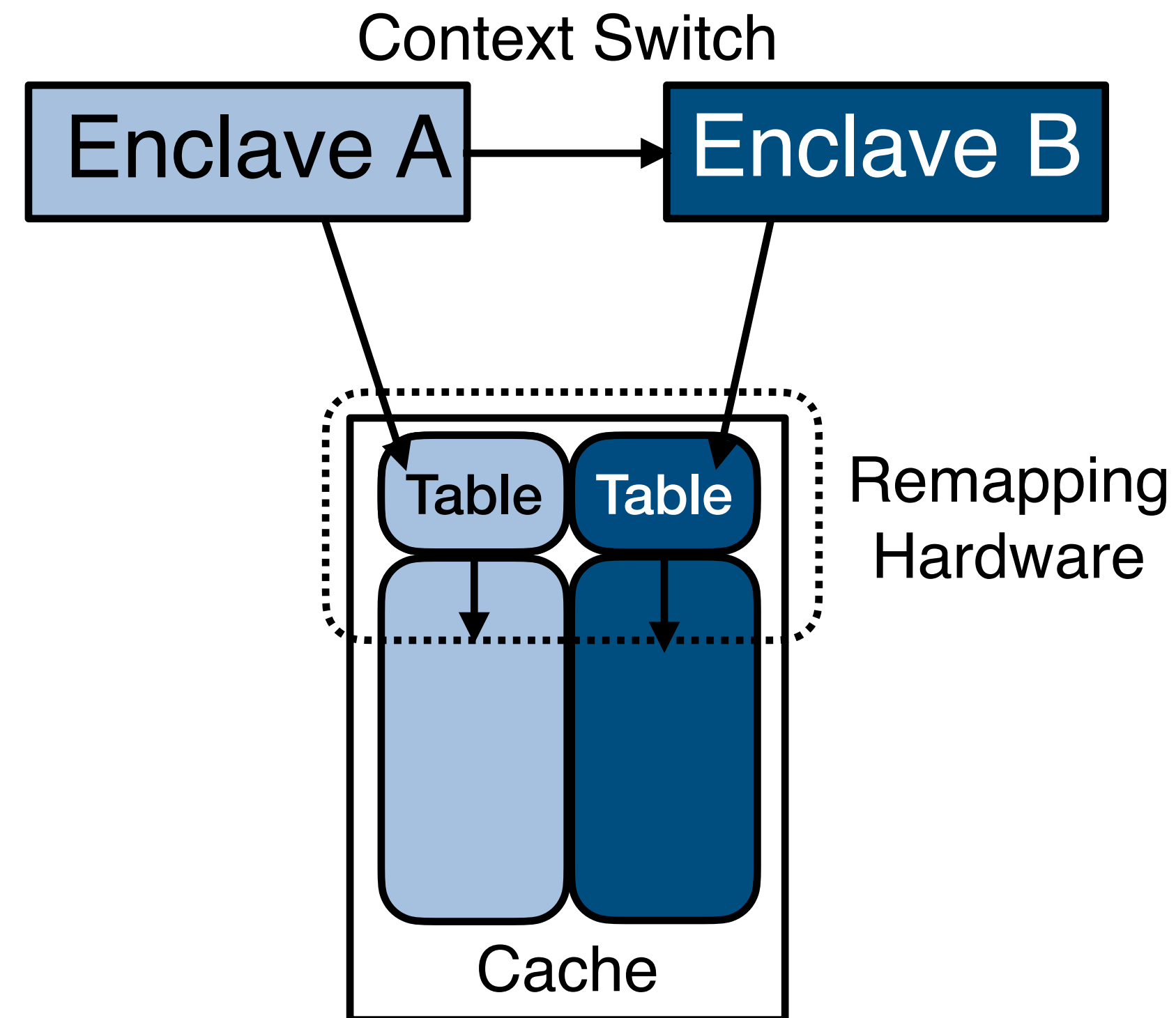
How to secure all cache levels without system software support?

- **Scalability:** support high number of enclaves with minimal hardware
- Context switches
- Cache coherence
- Provable security

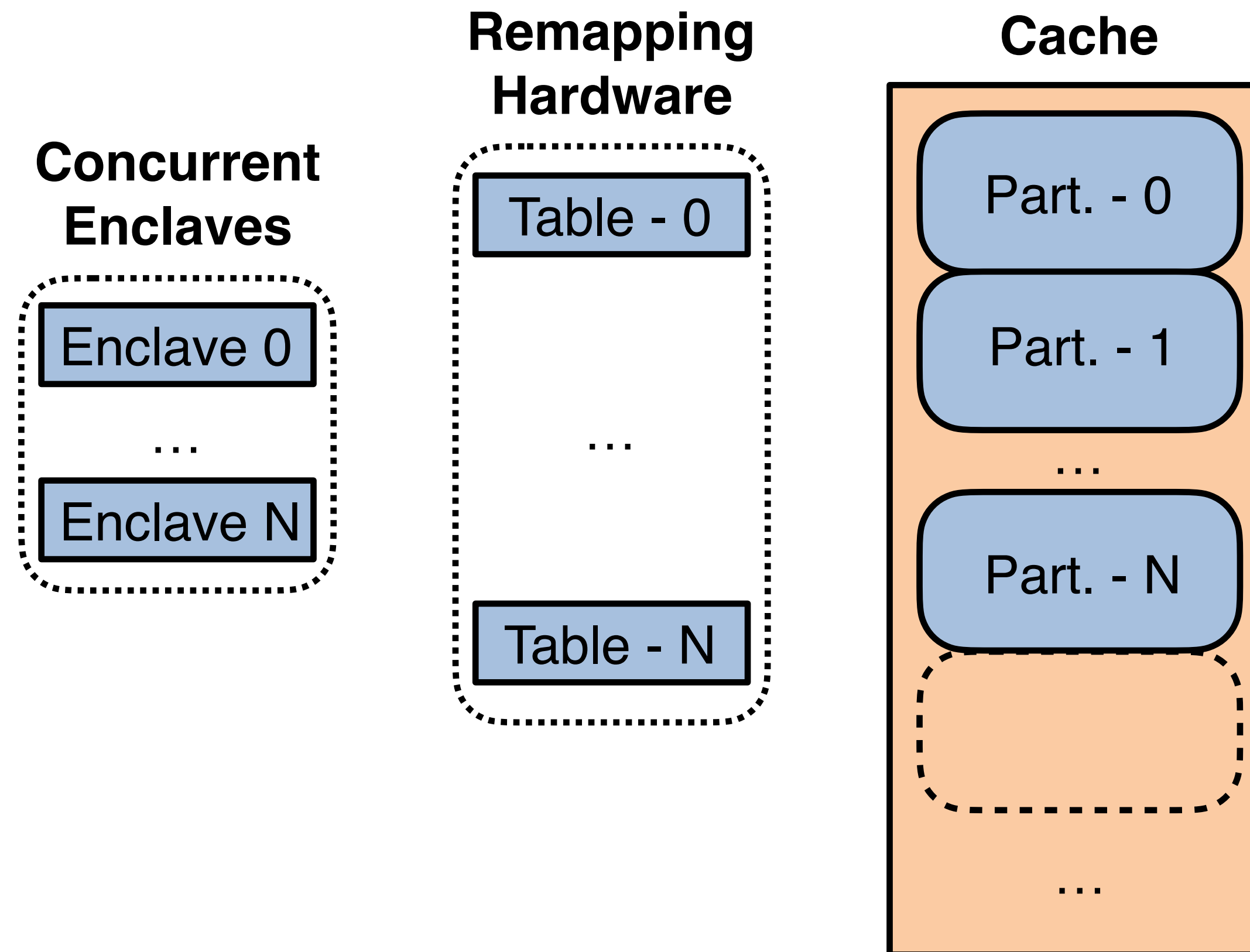


Why is Scalability Challenging?

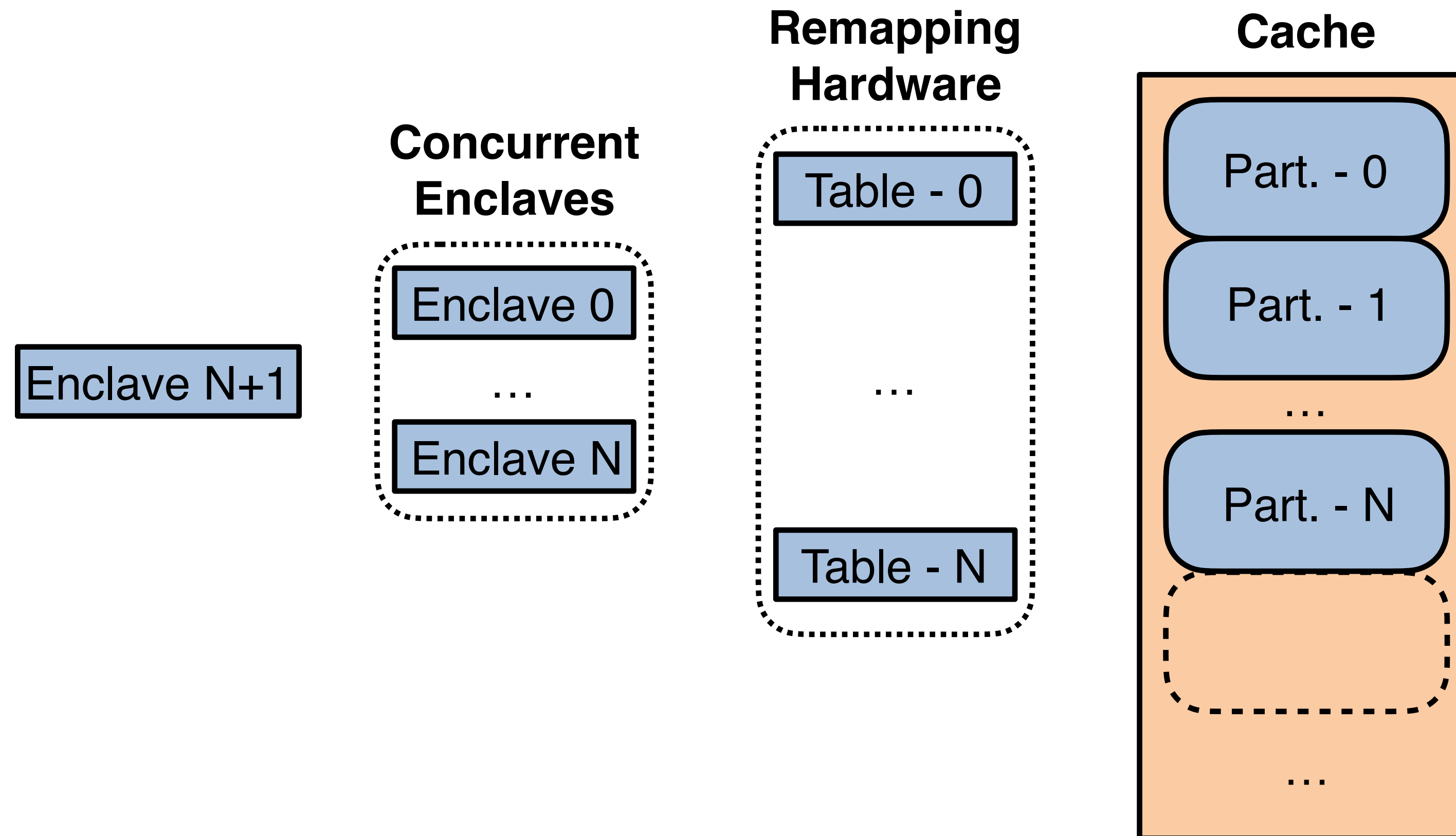
- Partitioning requires additional metadata associated with each partition:



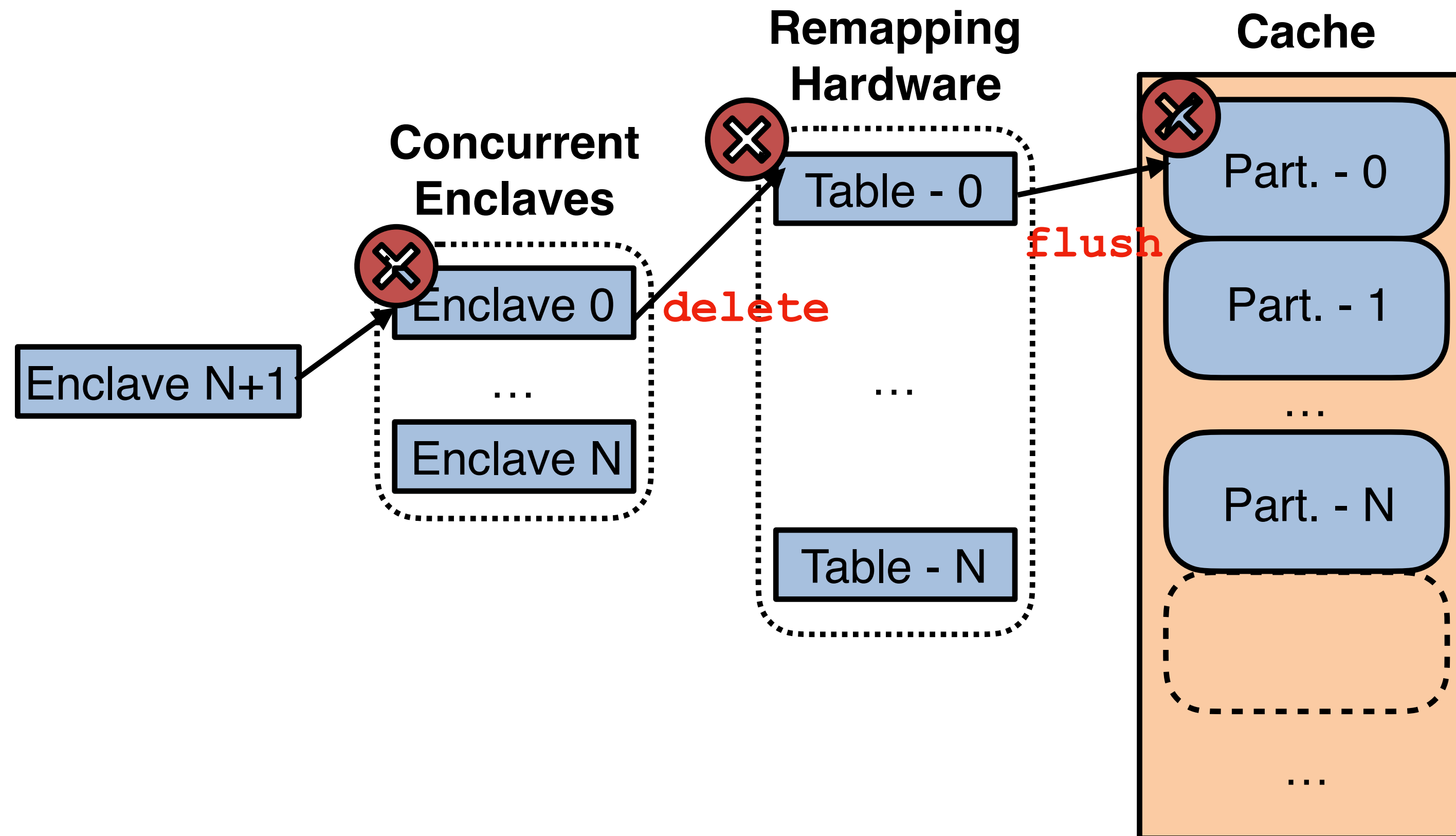
How **Not** to Manage Partition Metadata



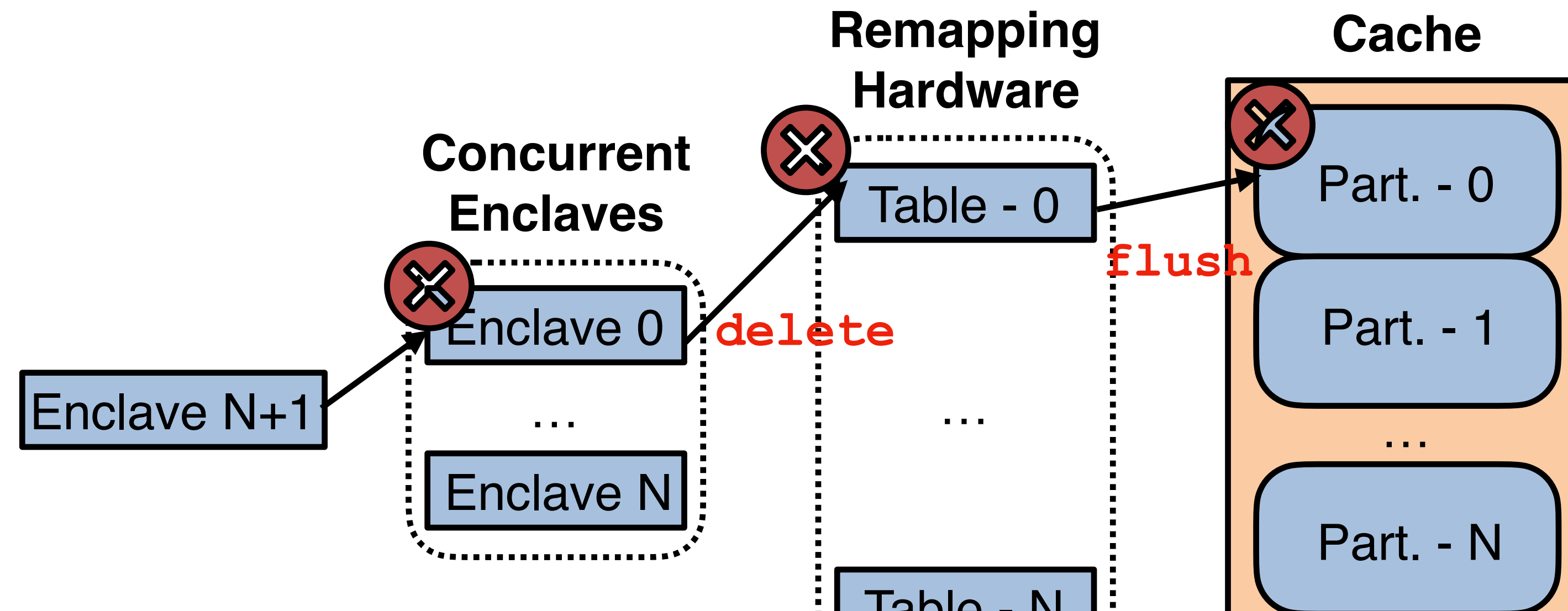
How **Not** to Manage Partition Metadata



How **Not** to Manage Partition Metadata

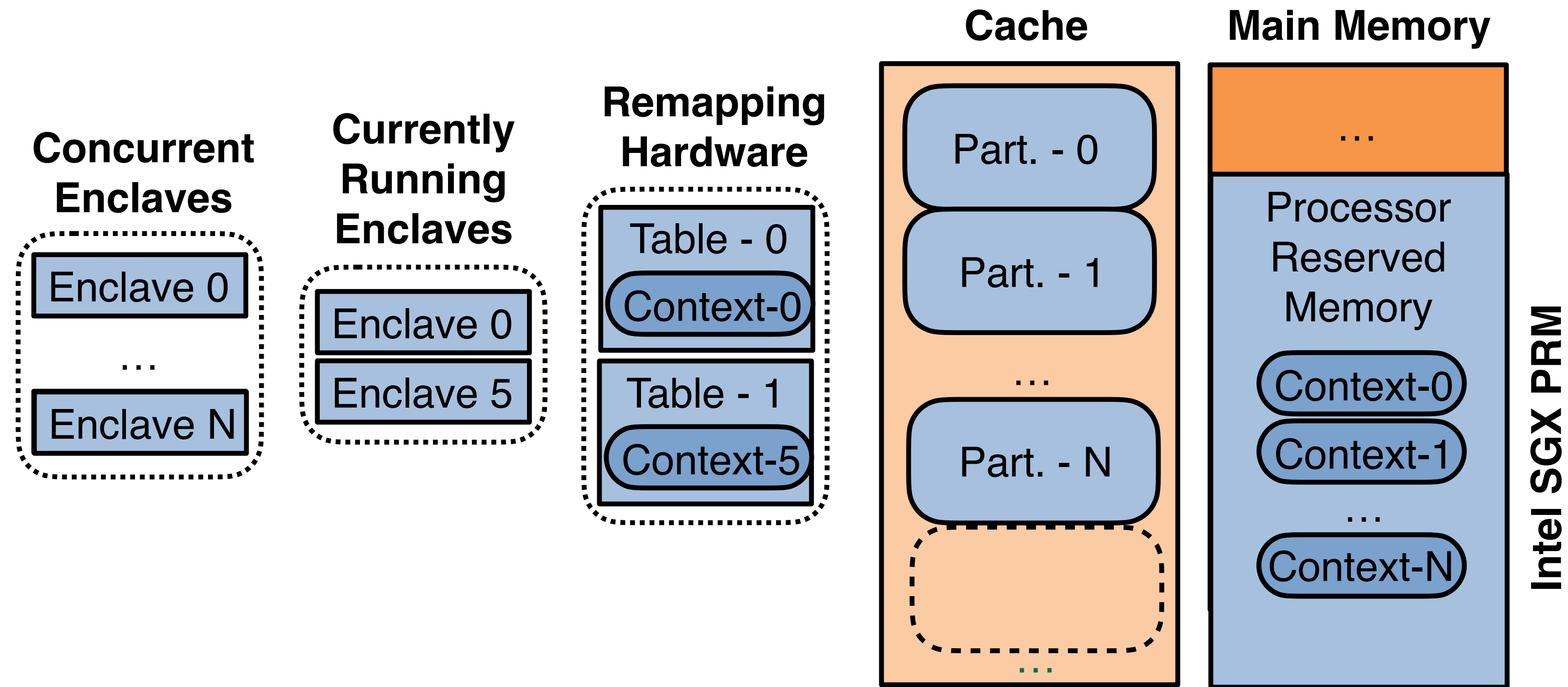


How **Not** to Manage Partition Metadata

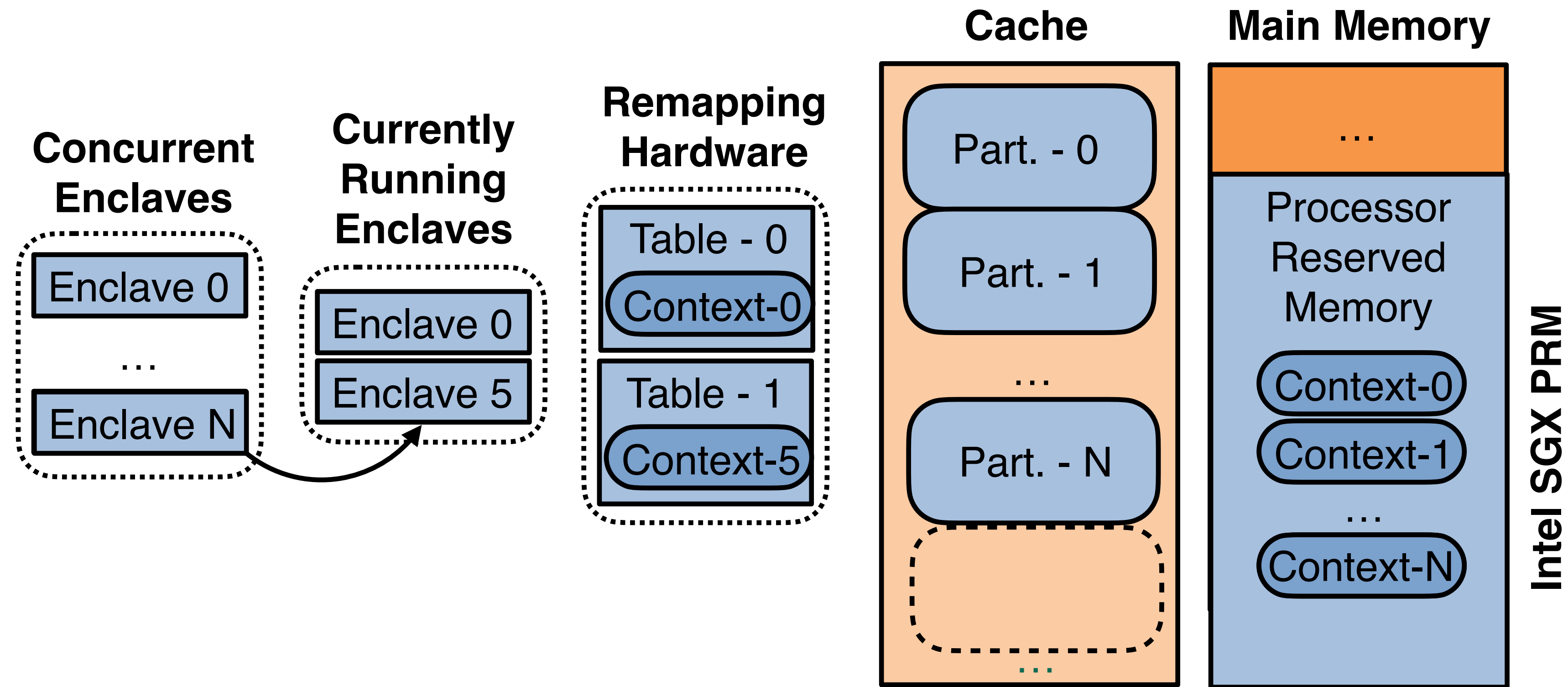


Problem: the maximum number of concurrent enclaves is limited by the number of tables provided by the remapping logic.

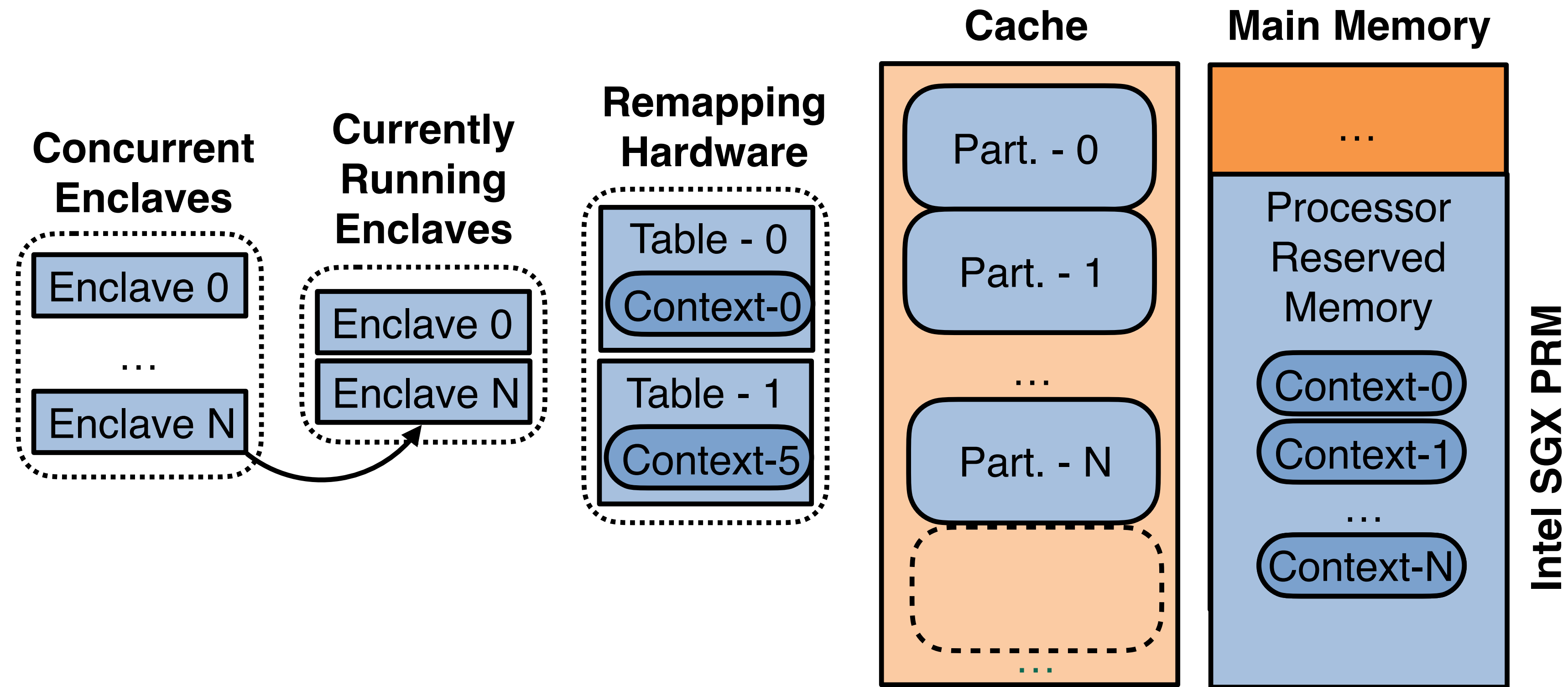
TEE-ShirT Metadata Virtualization



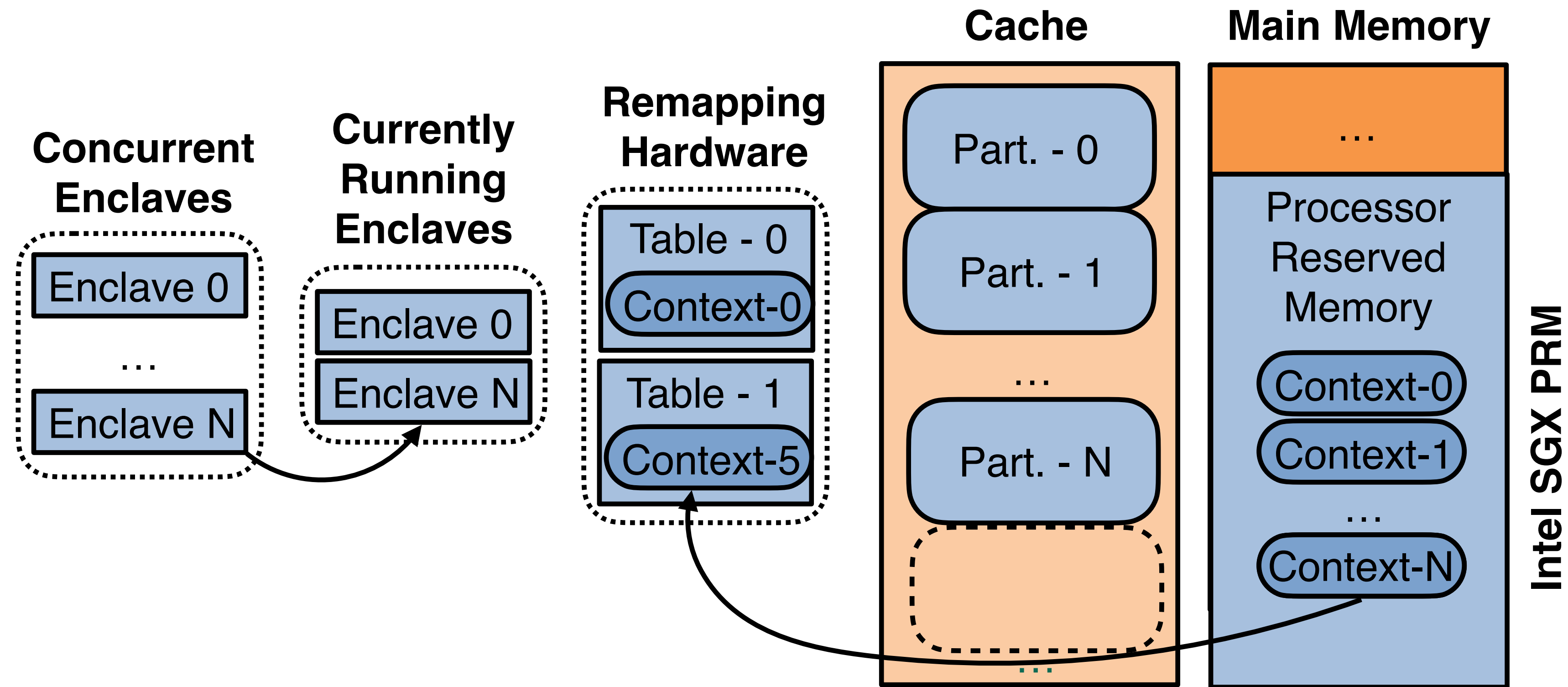
TEE-ShirT Metadata Virtualization



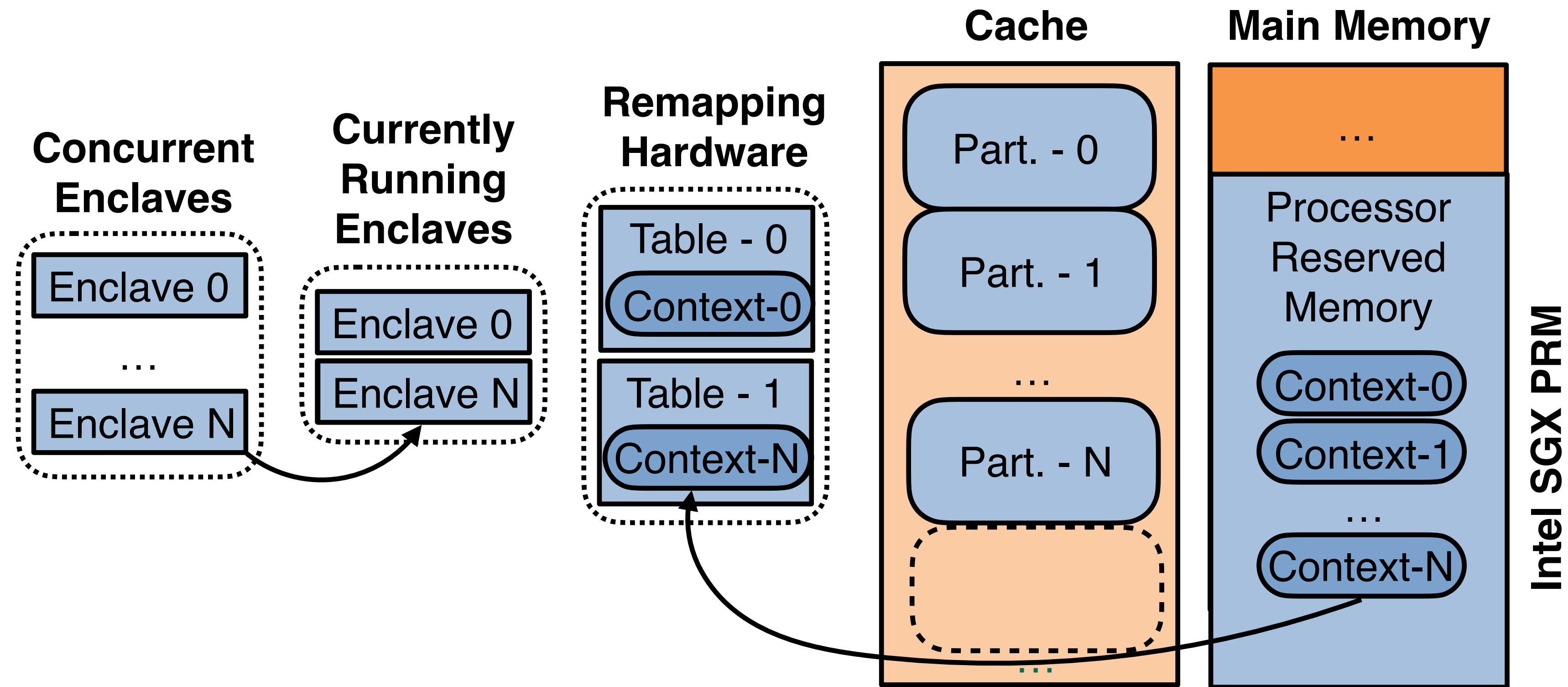
TEE-ShirT Metadata Virtualization



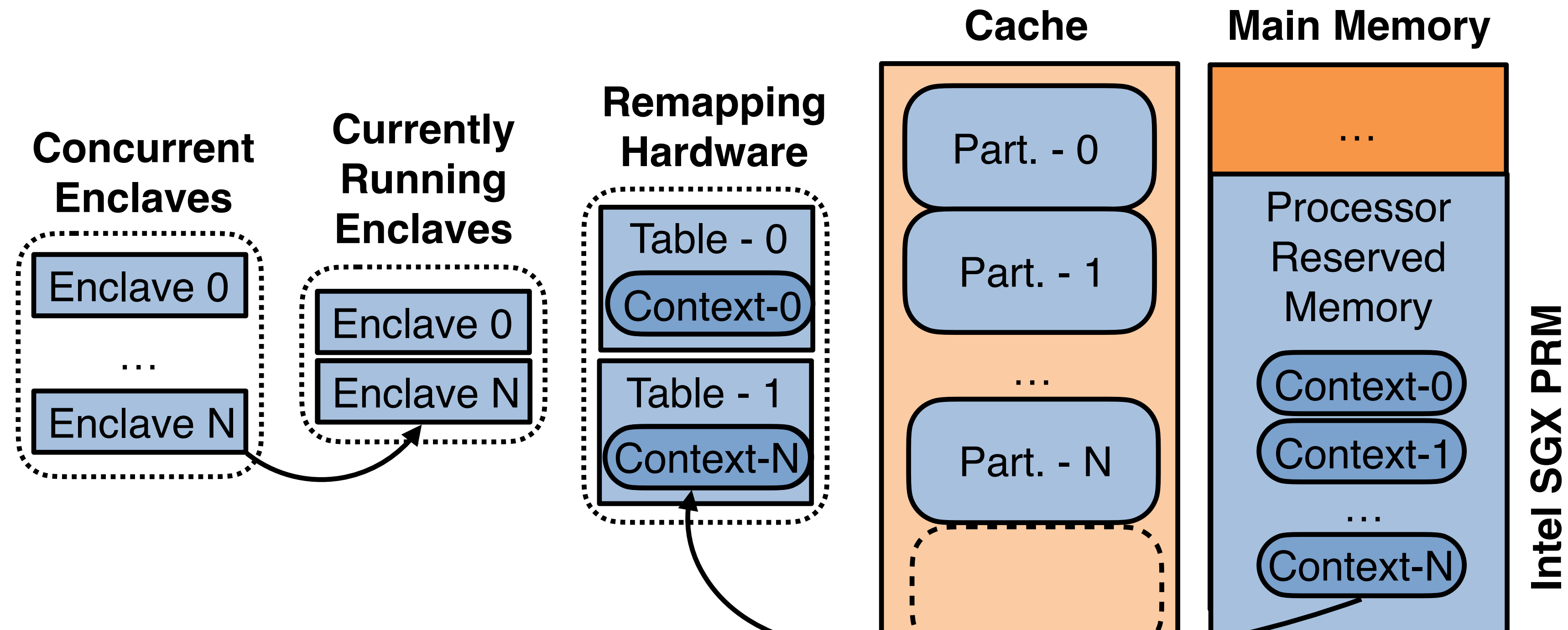
TEE-ShirT Metadata Virtualization



TEE-ShirT Metadata Virtualization



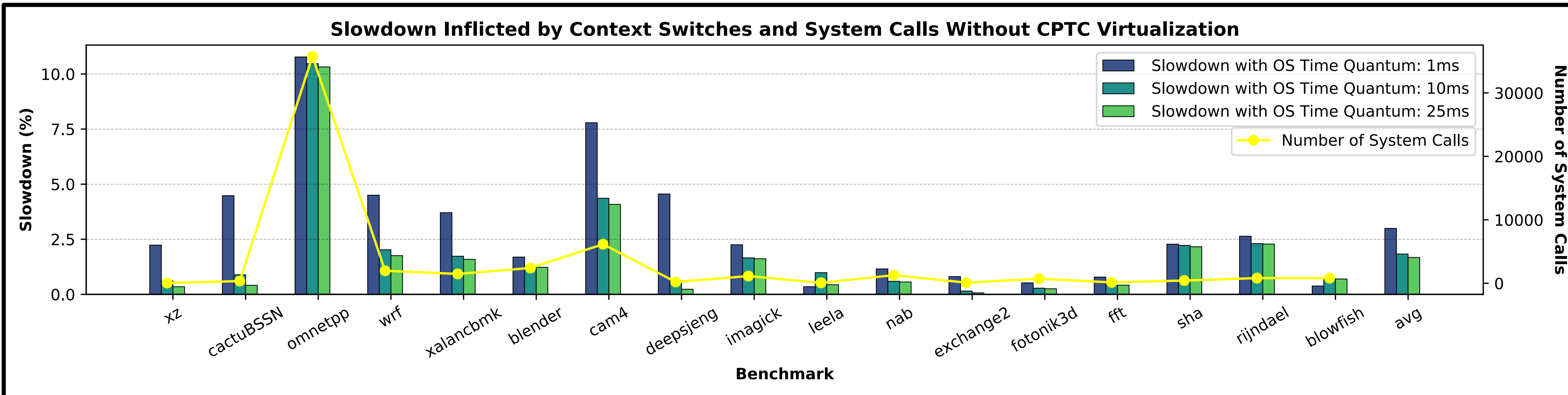
TEE-ShirT Metadata Virtualization



TEE-ShirT alleviates the context switch and system call time penalty and ensures lower hardware complexity.

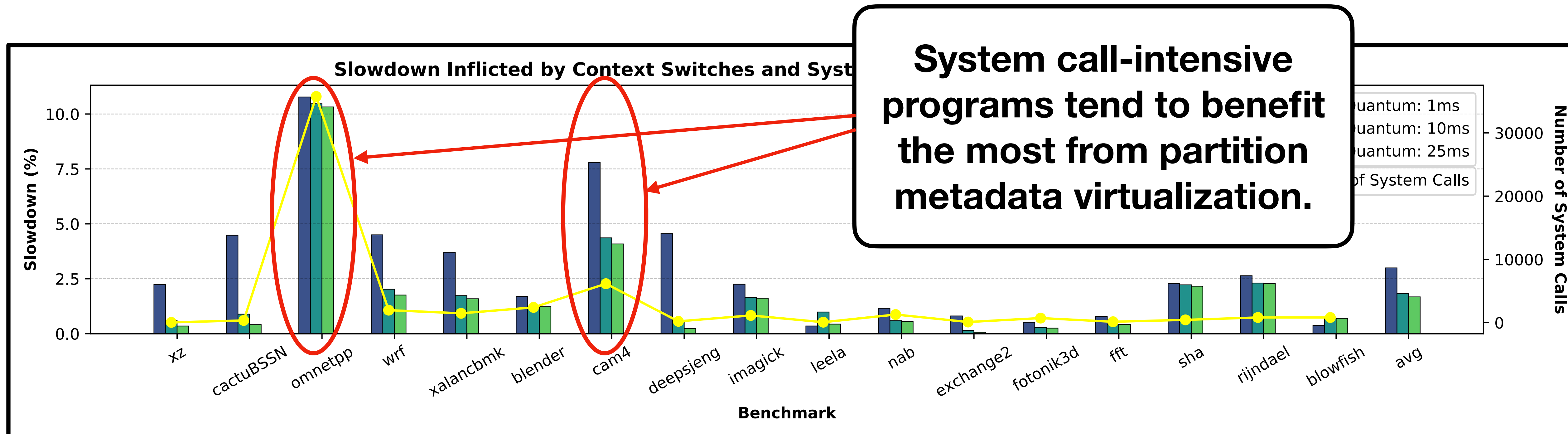
Metadata Virtualization: Results

- Slowdown inflicted by cache flushes is greatly alleviated by metadata virtualization:



Metadata Virtualization: Results

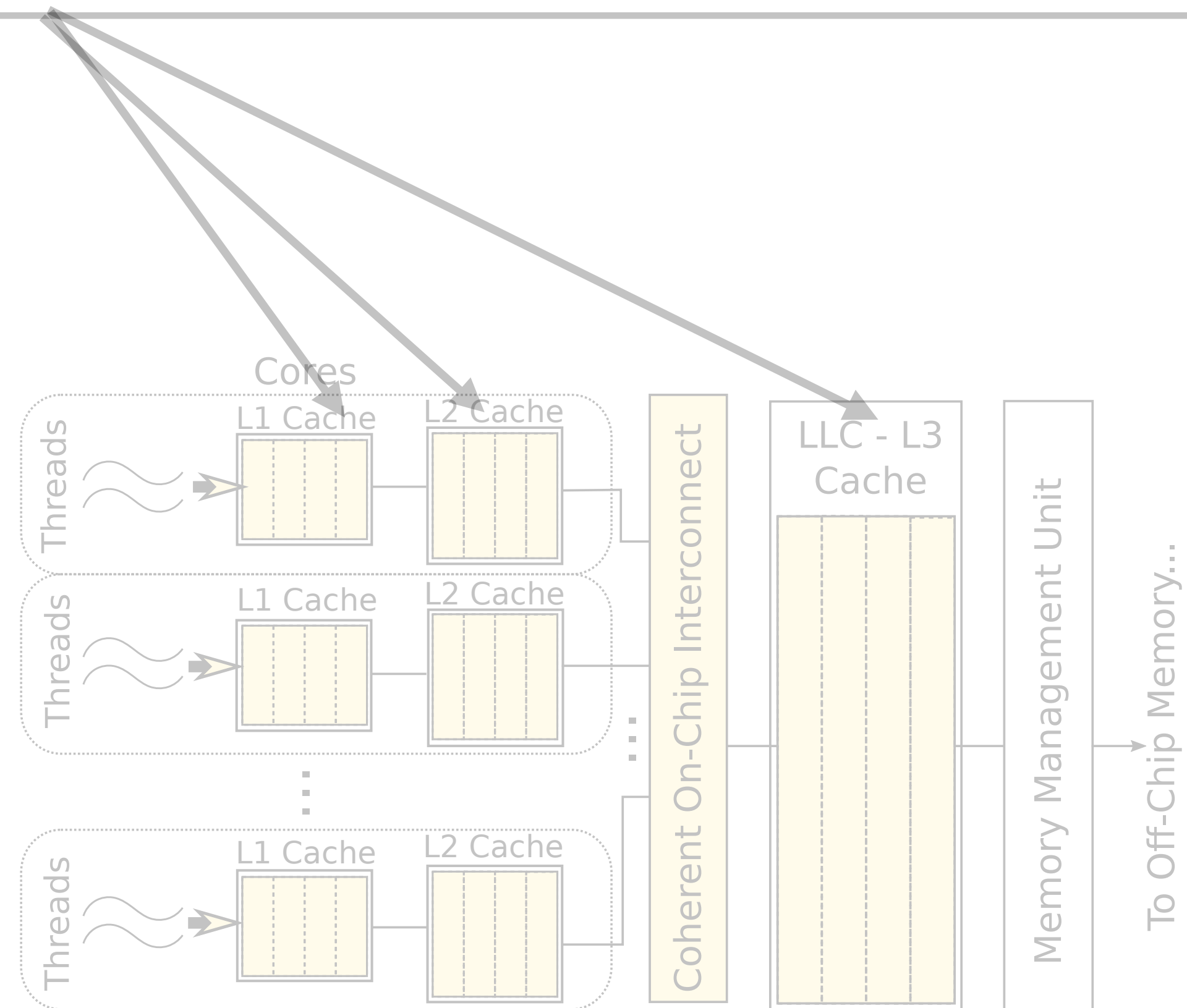
- Slowdown inflicted by cache flushes is greatly alleviated by metadata virtualization:



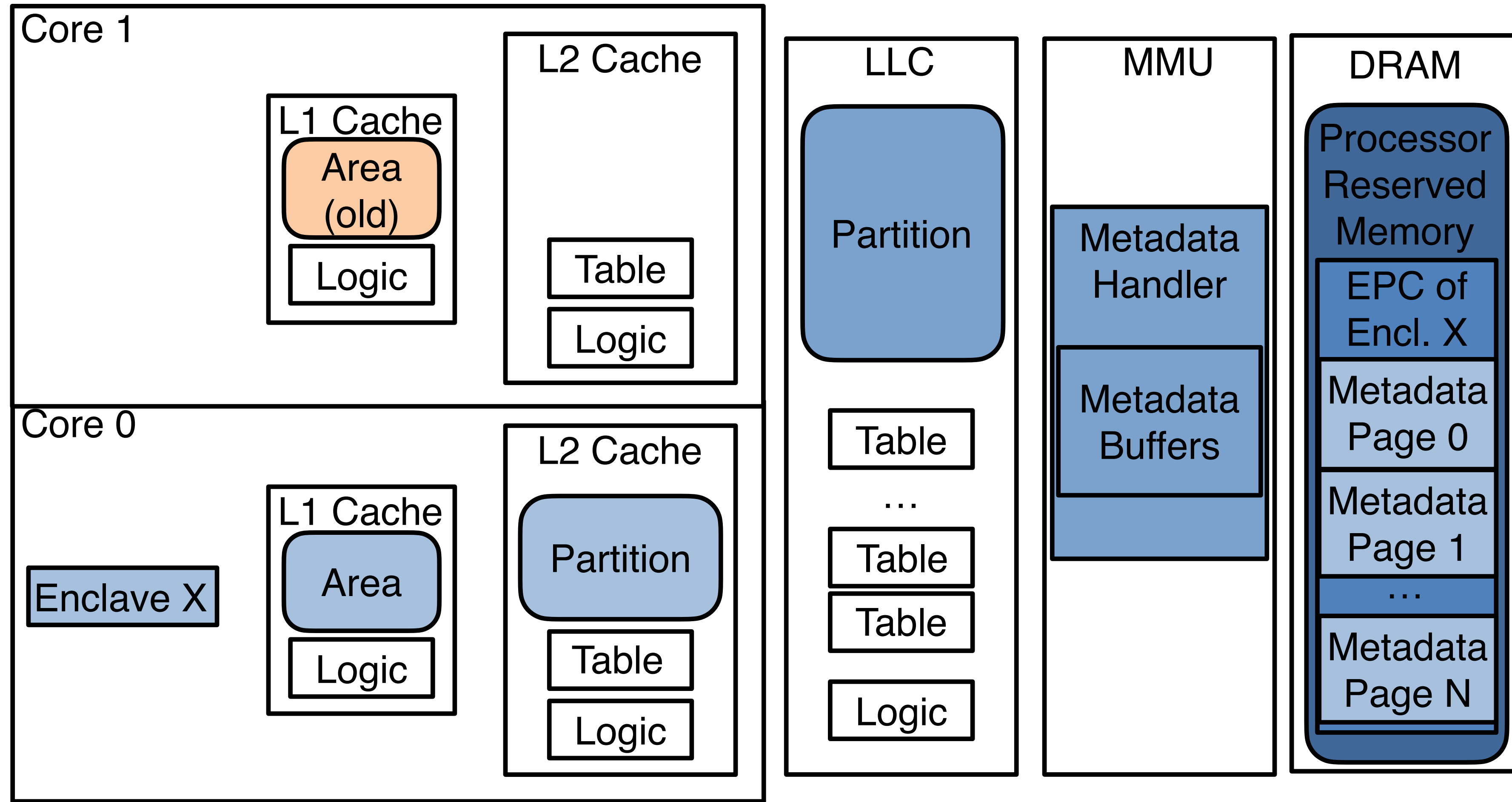
This Talk: TEE-ShirT

How to secure all cache levels without system software support?

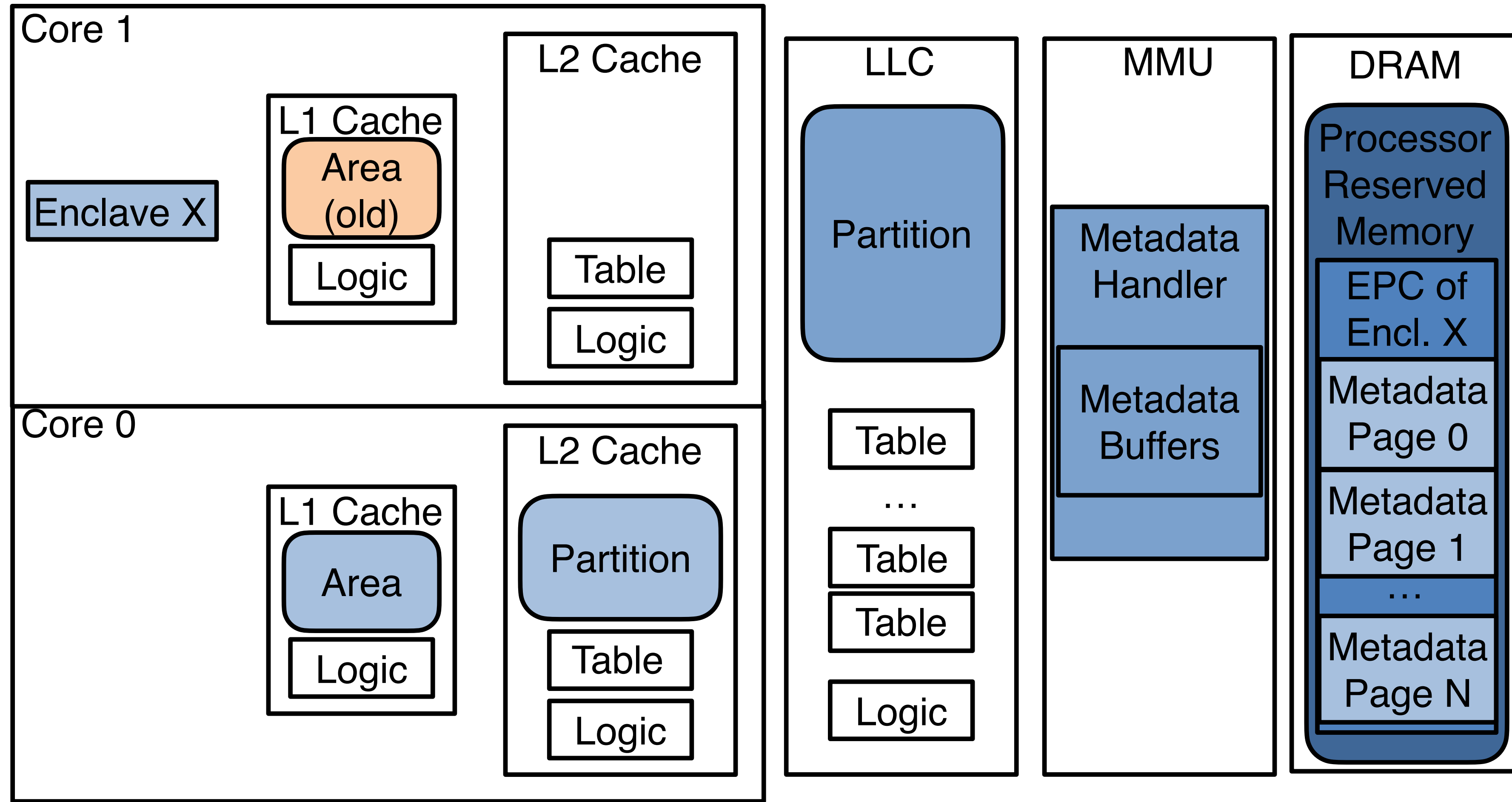
- **Scalability:** support high number of enclaves with minimal hardware
- **Context switches**
- **Cache coherence**
- **Provable security**



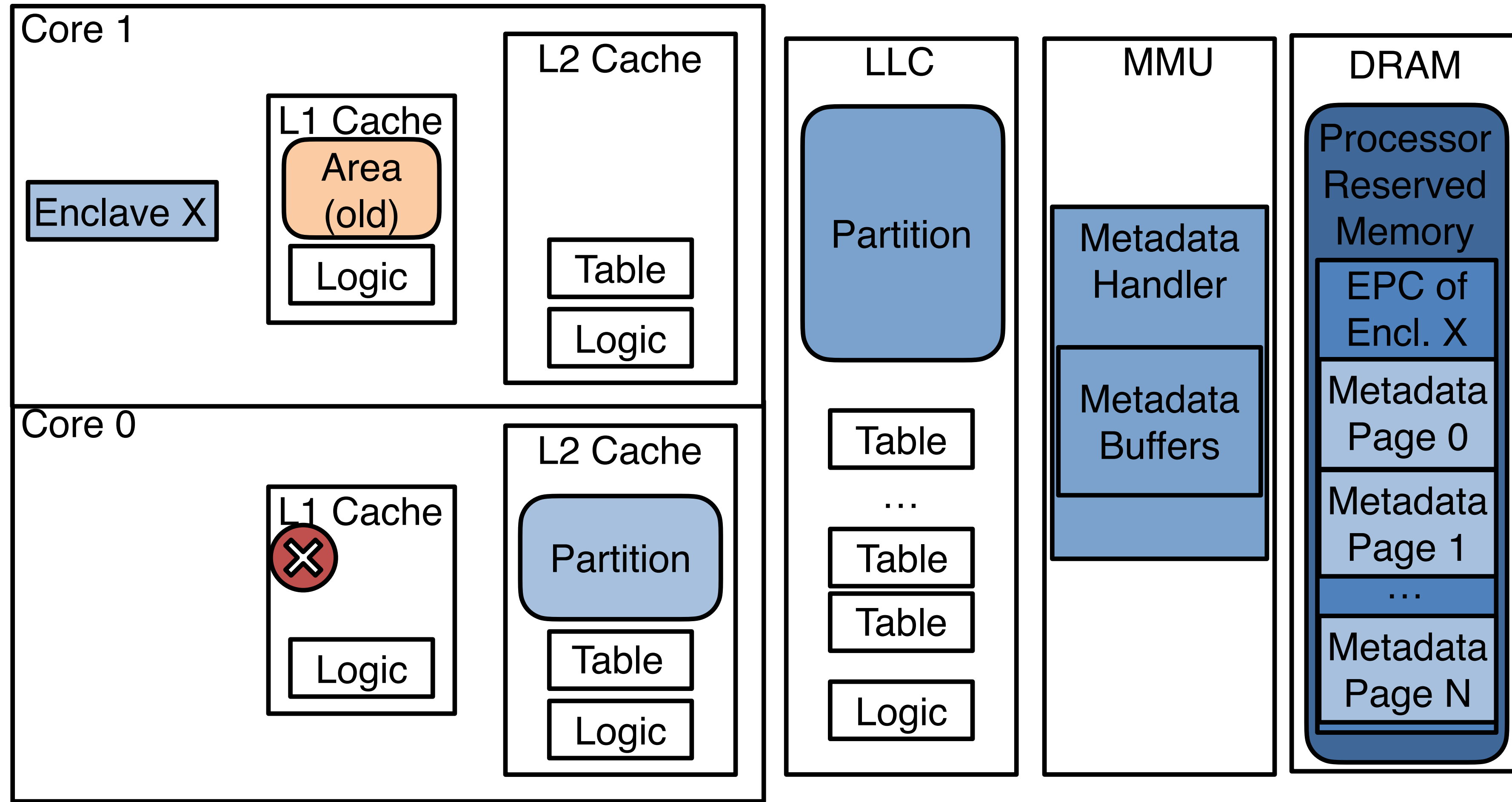
Cache-Aware Context Switches



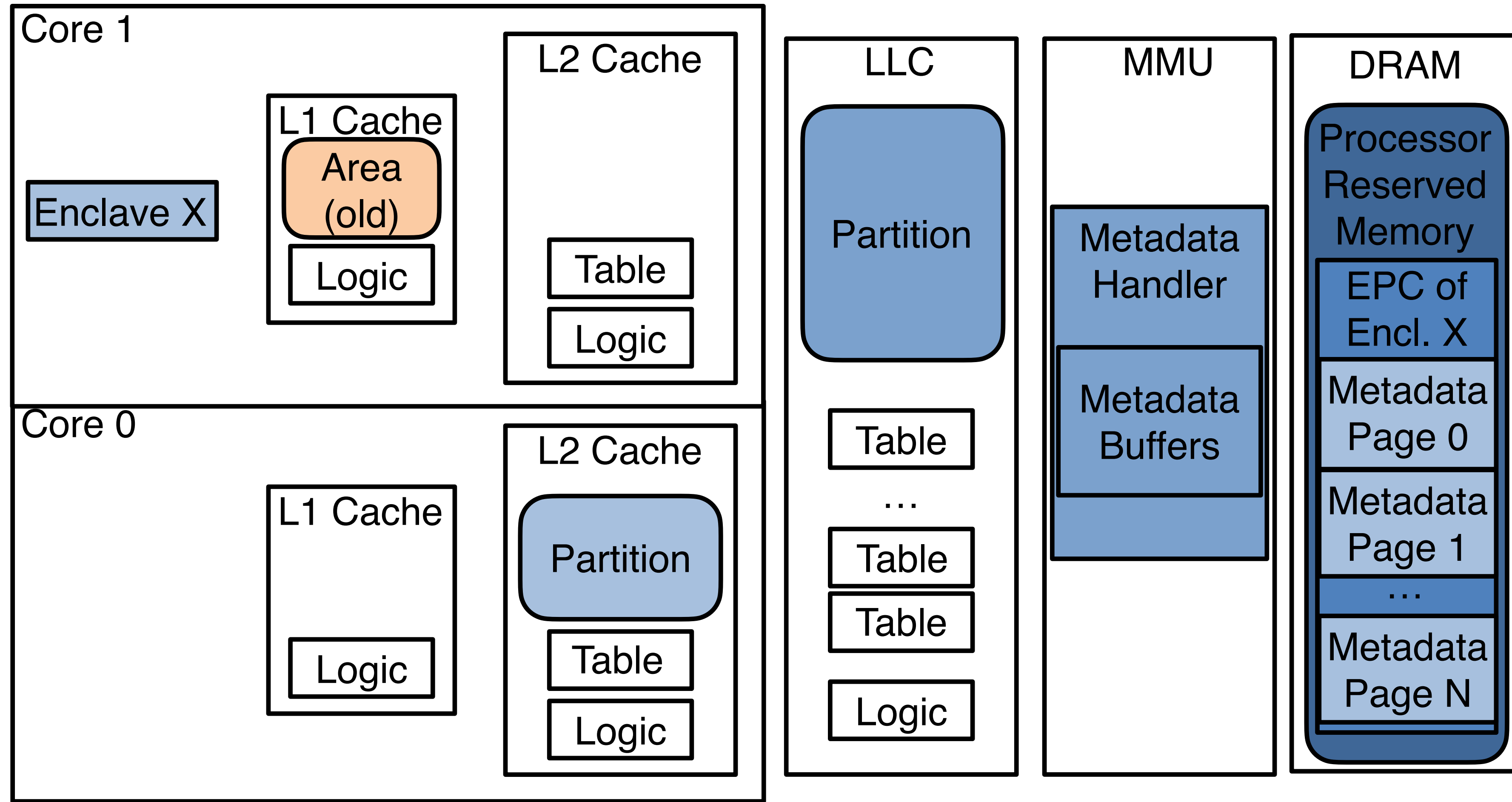
Cache-Aware Context Switches



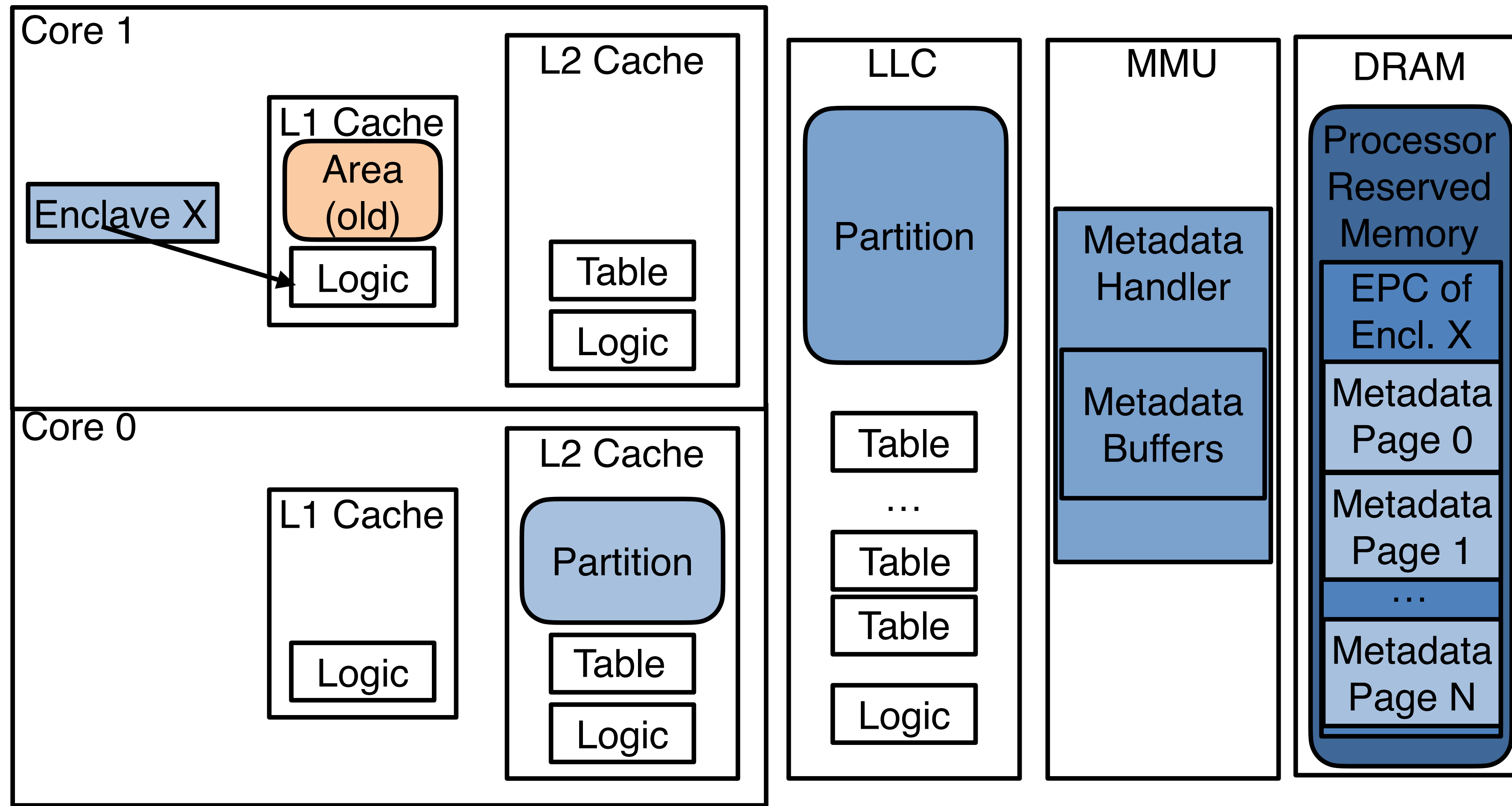
Cache-Aware Context Switches



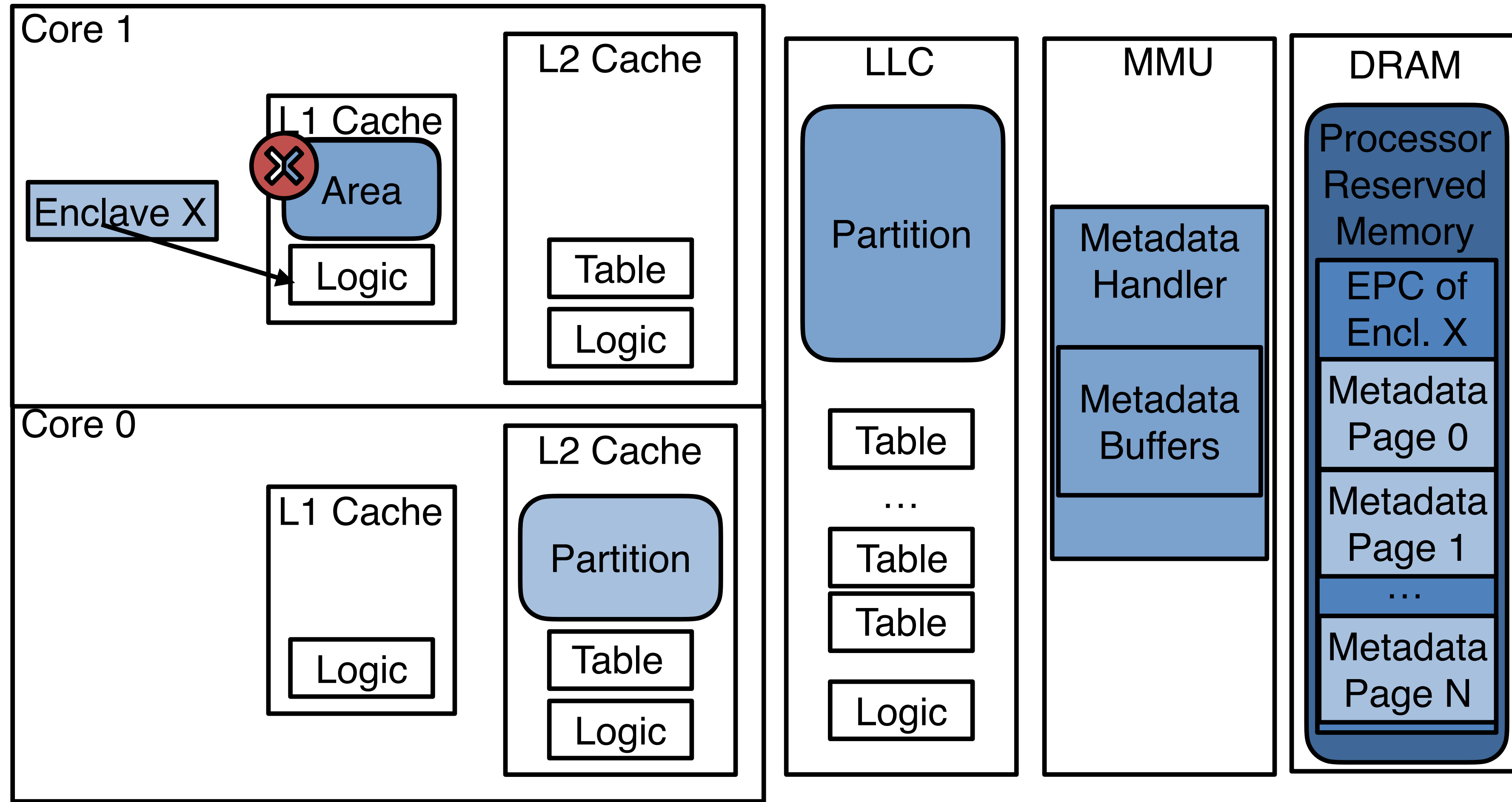
Cache-Aware Context Switches



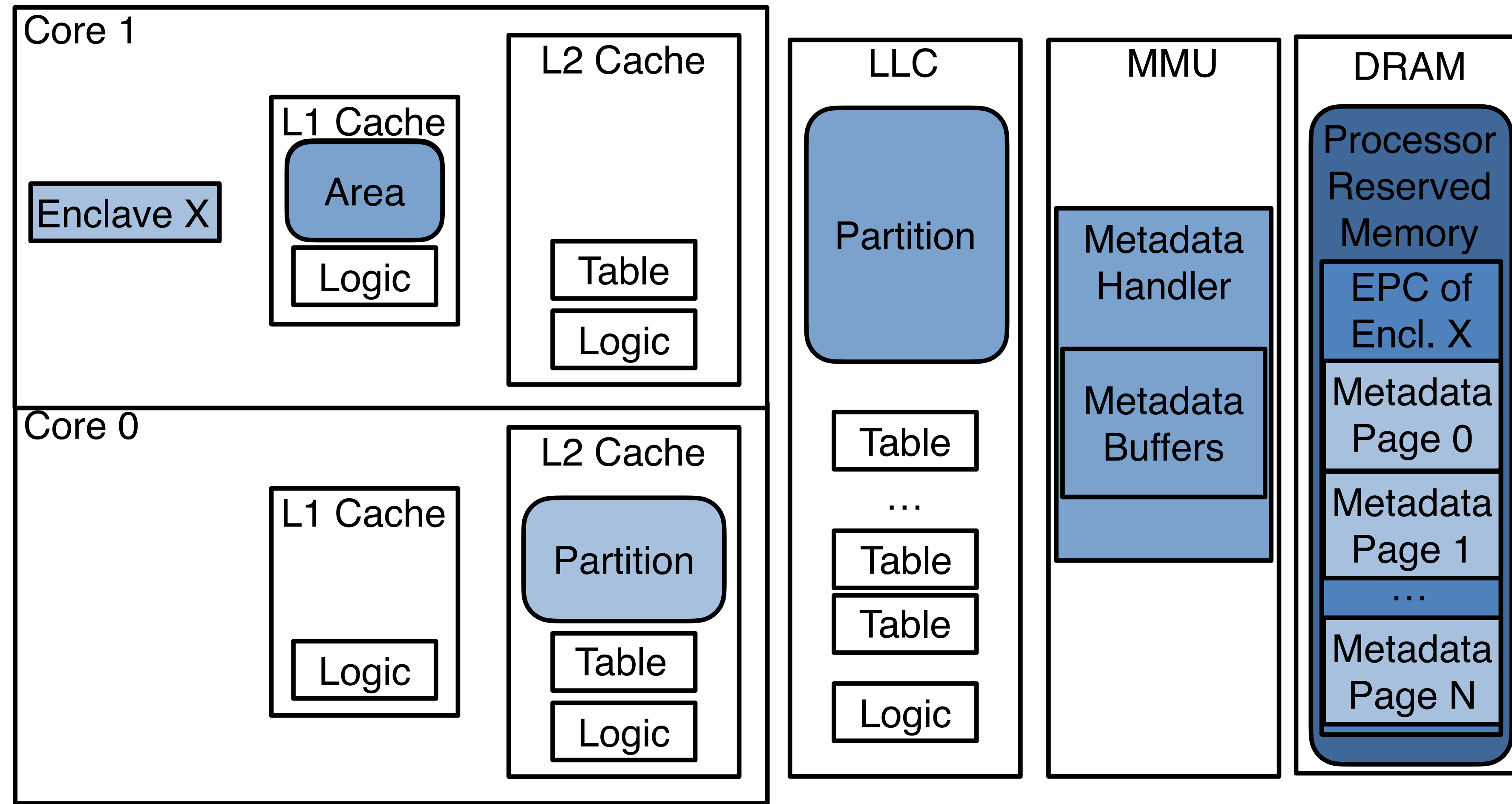
Cache-Aware Context Switches



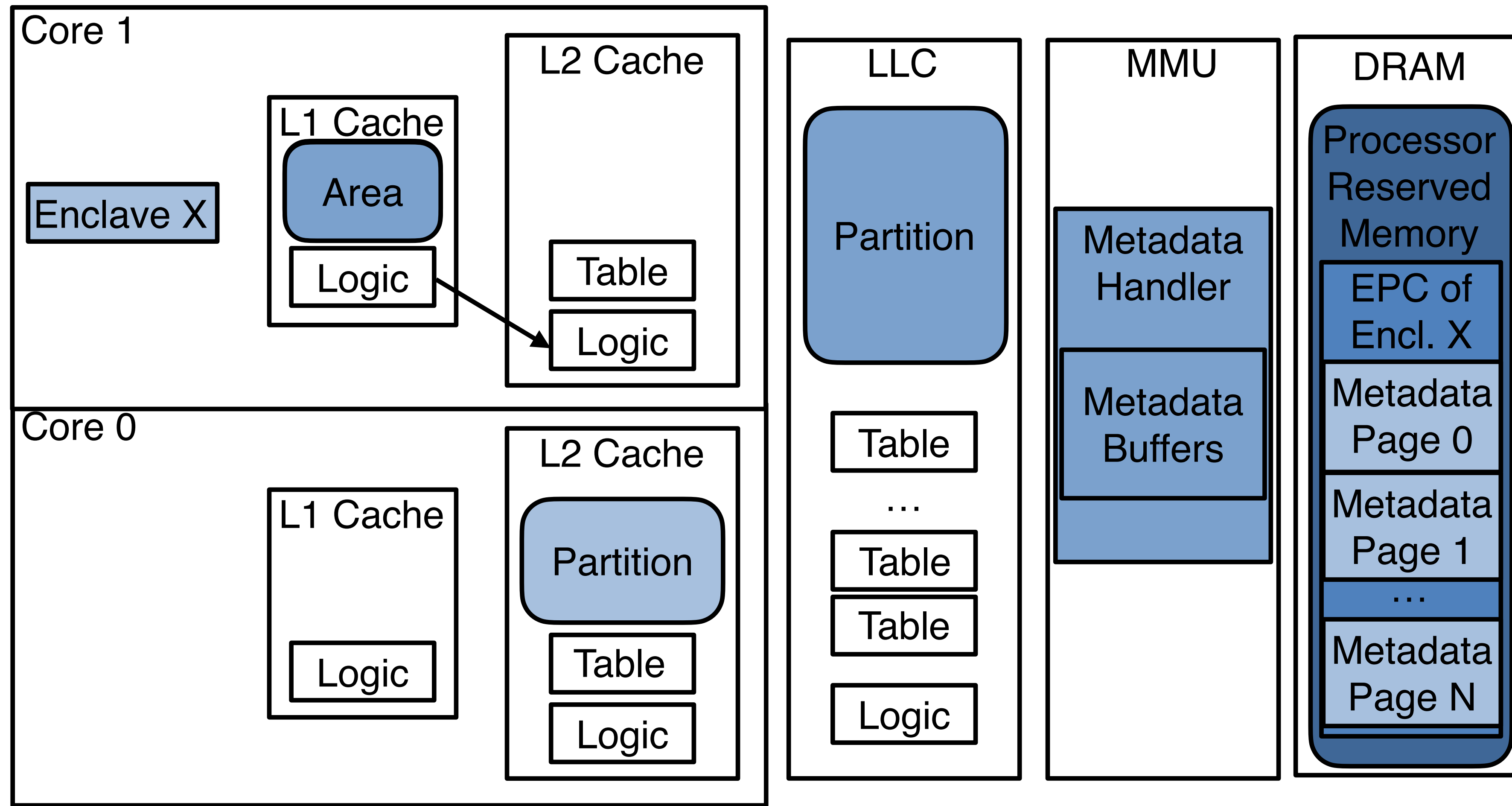
Cache-Aware Context Switches



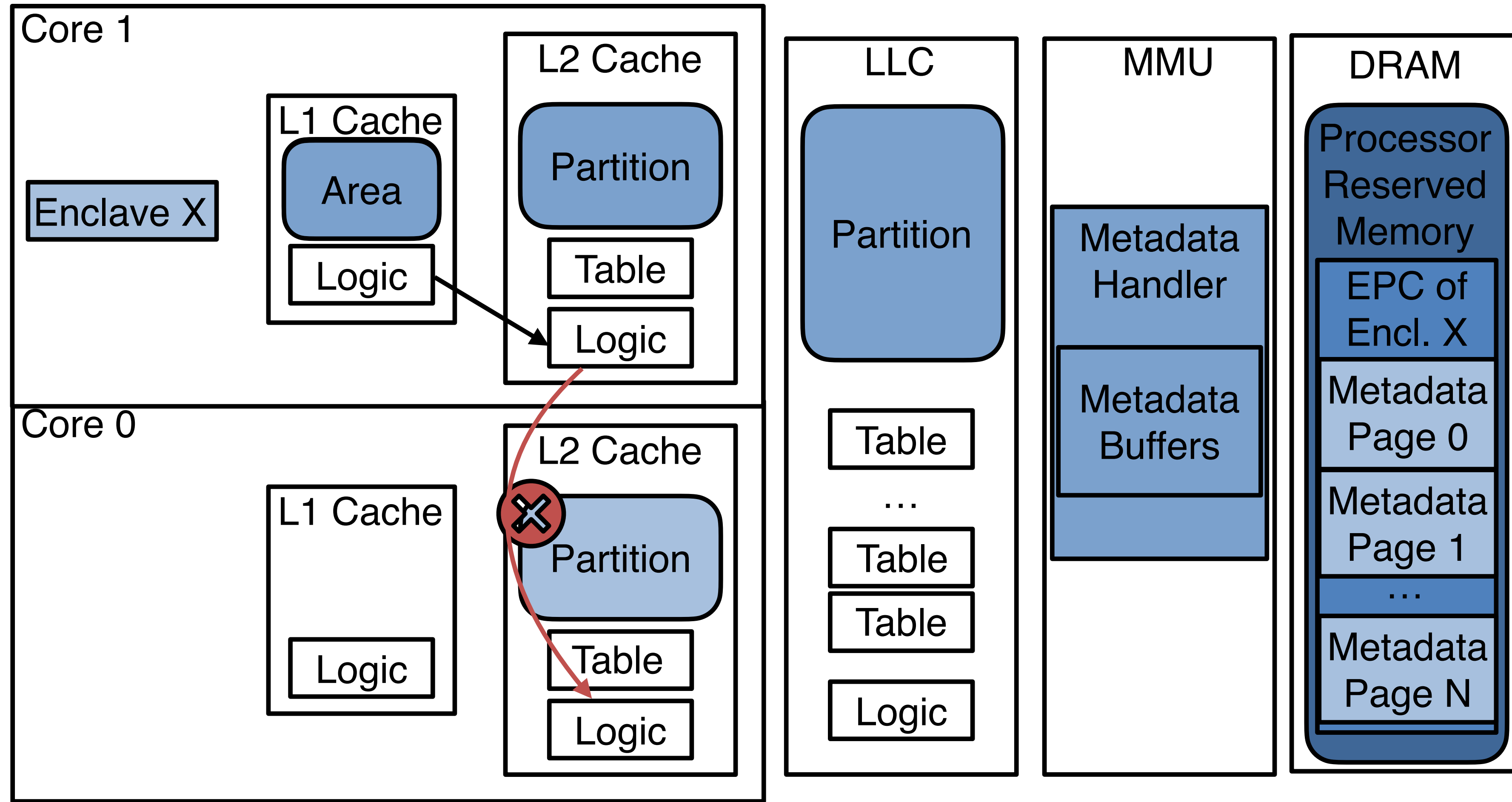
Cache-Aware Context Switches



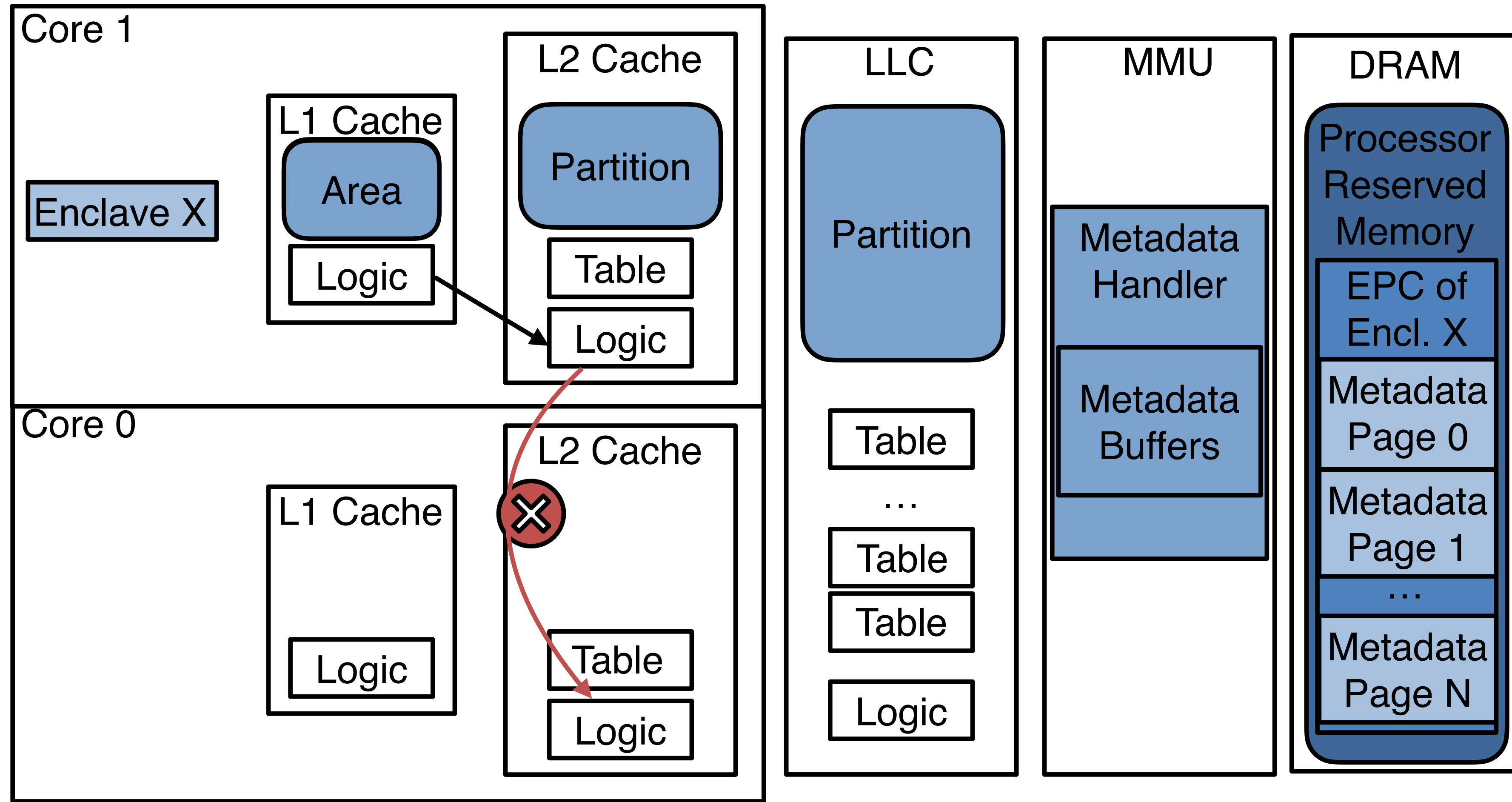
Cache-Aware Context Switches



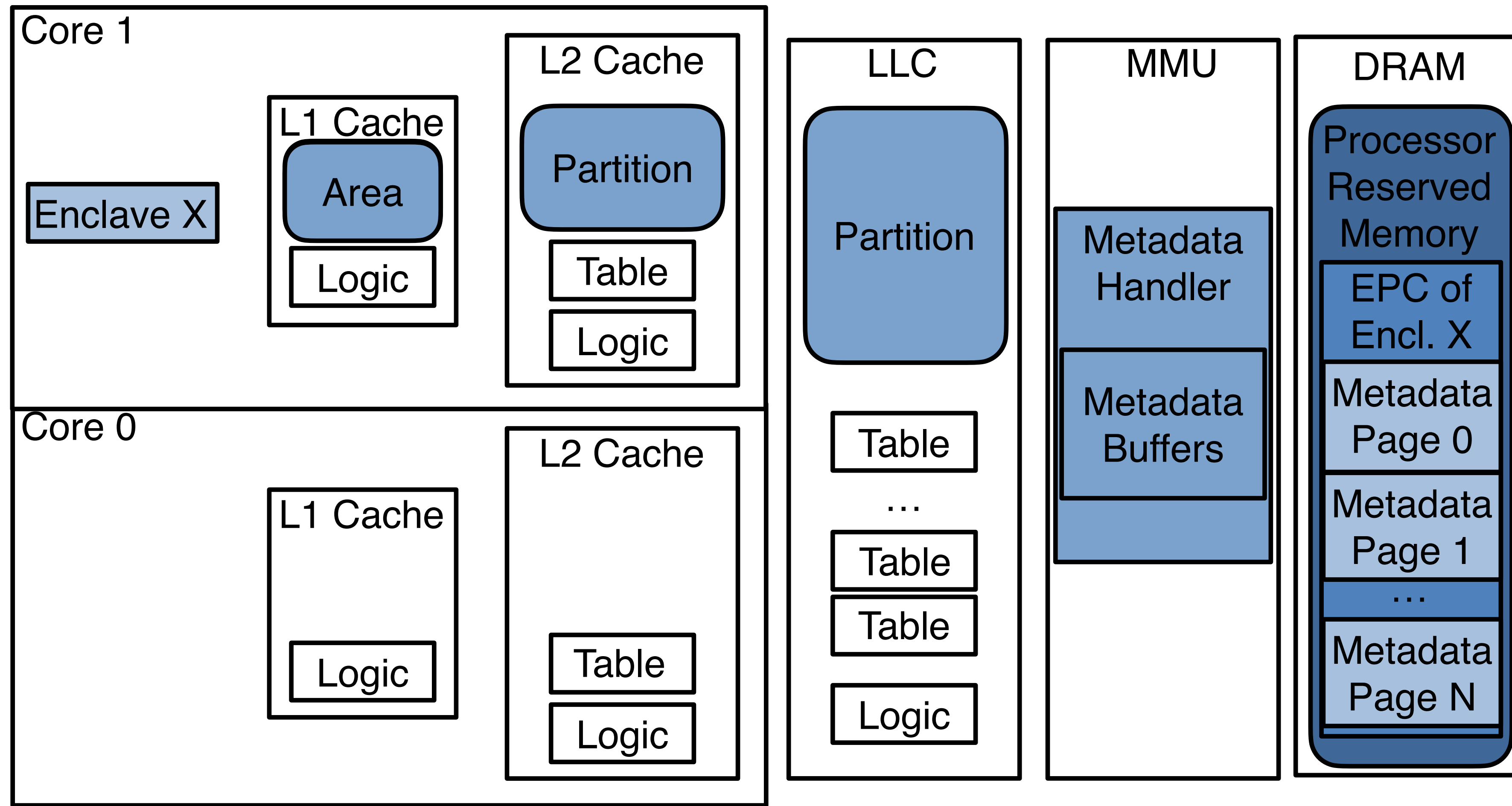
Cache-Aware Context Switches



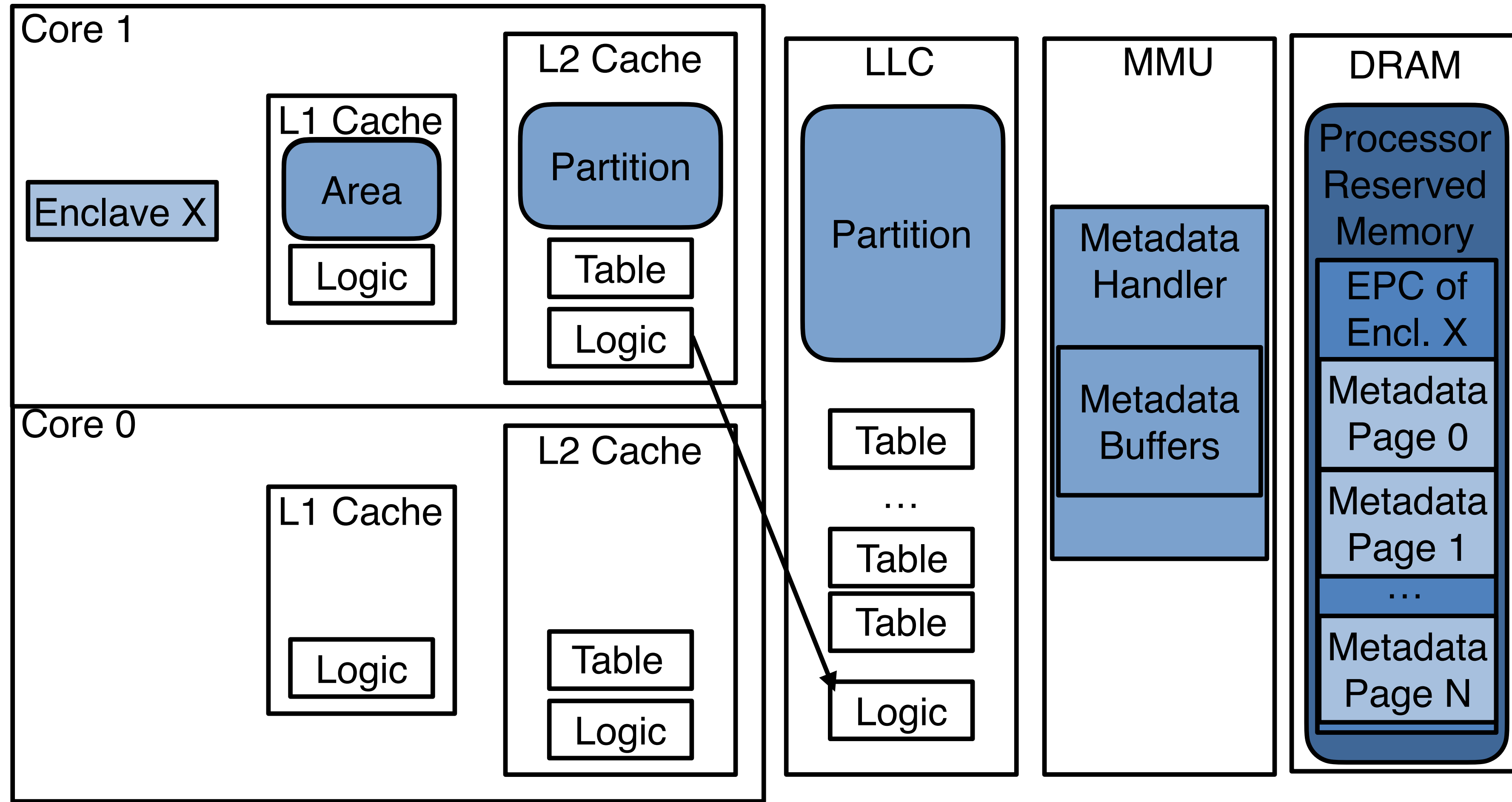
Cache-Aware Context Switches



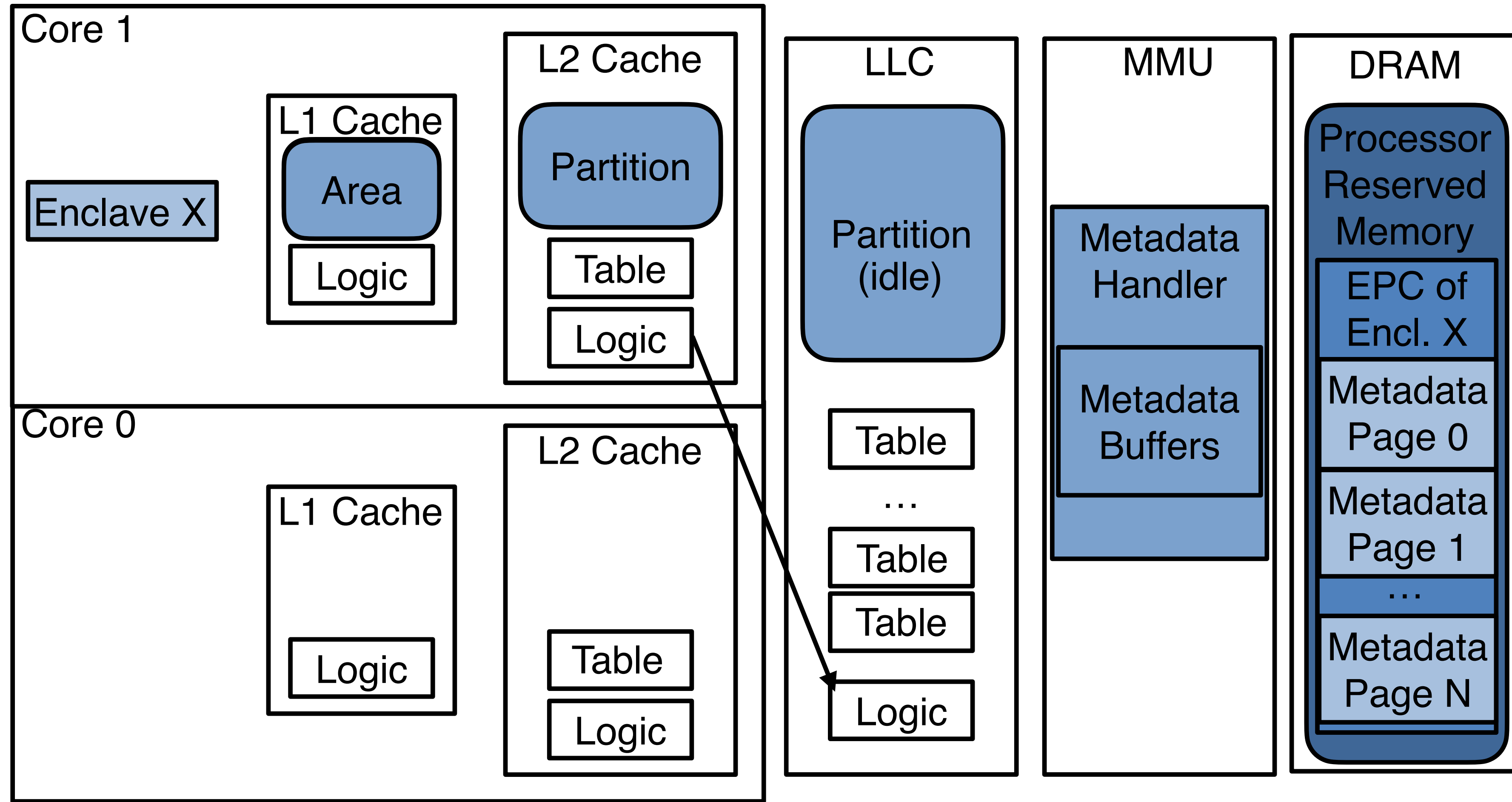
Cache-Aware Context Switches



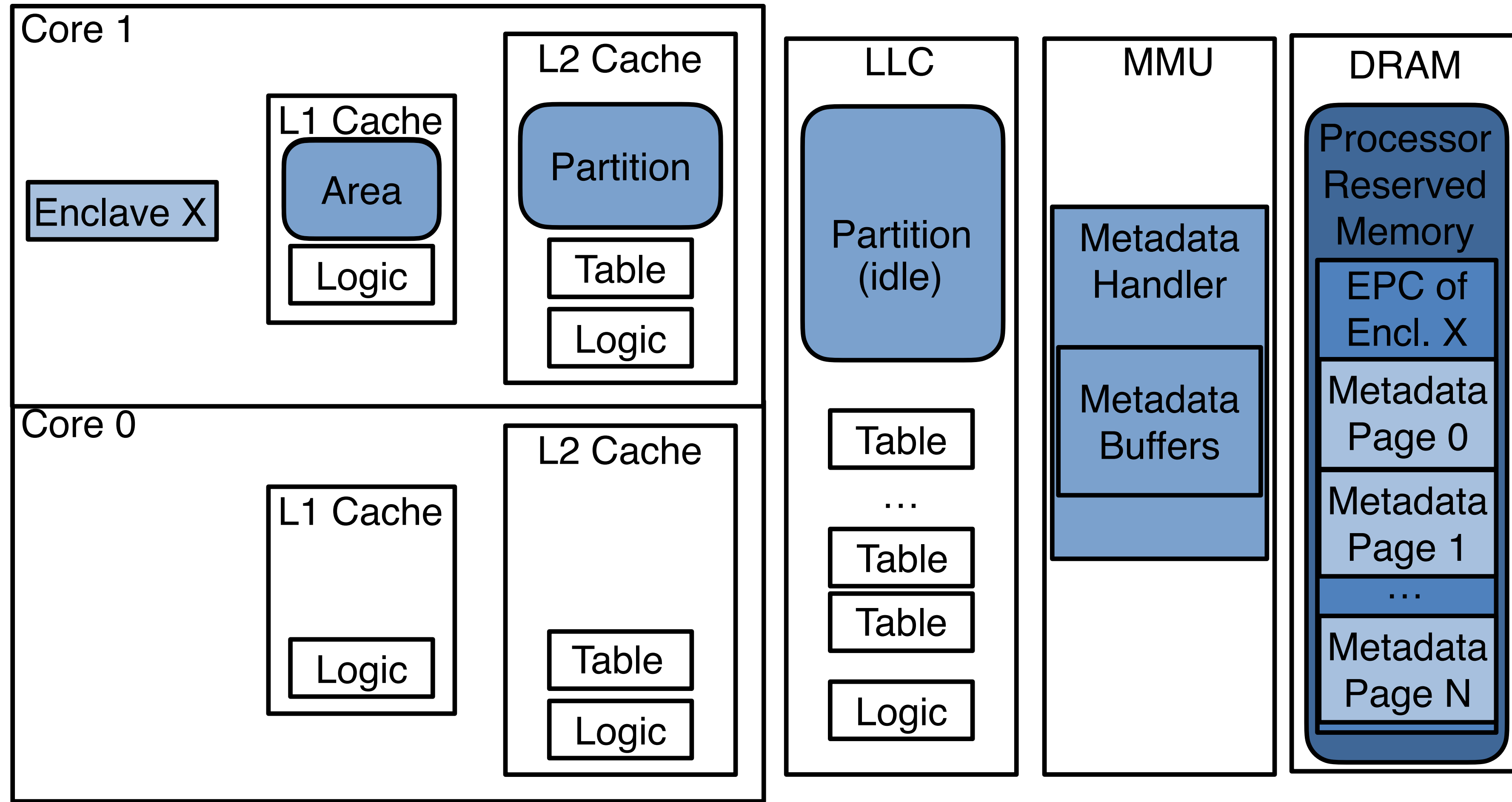
Cache-Aware Context Switches



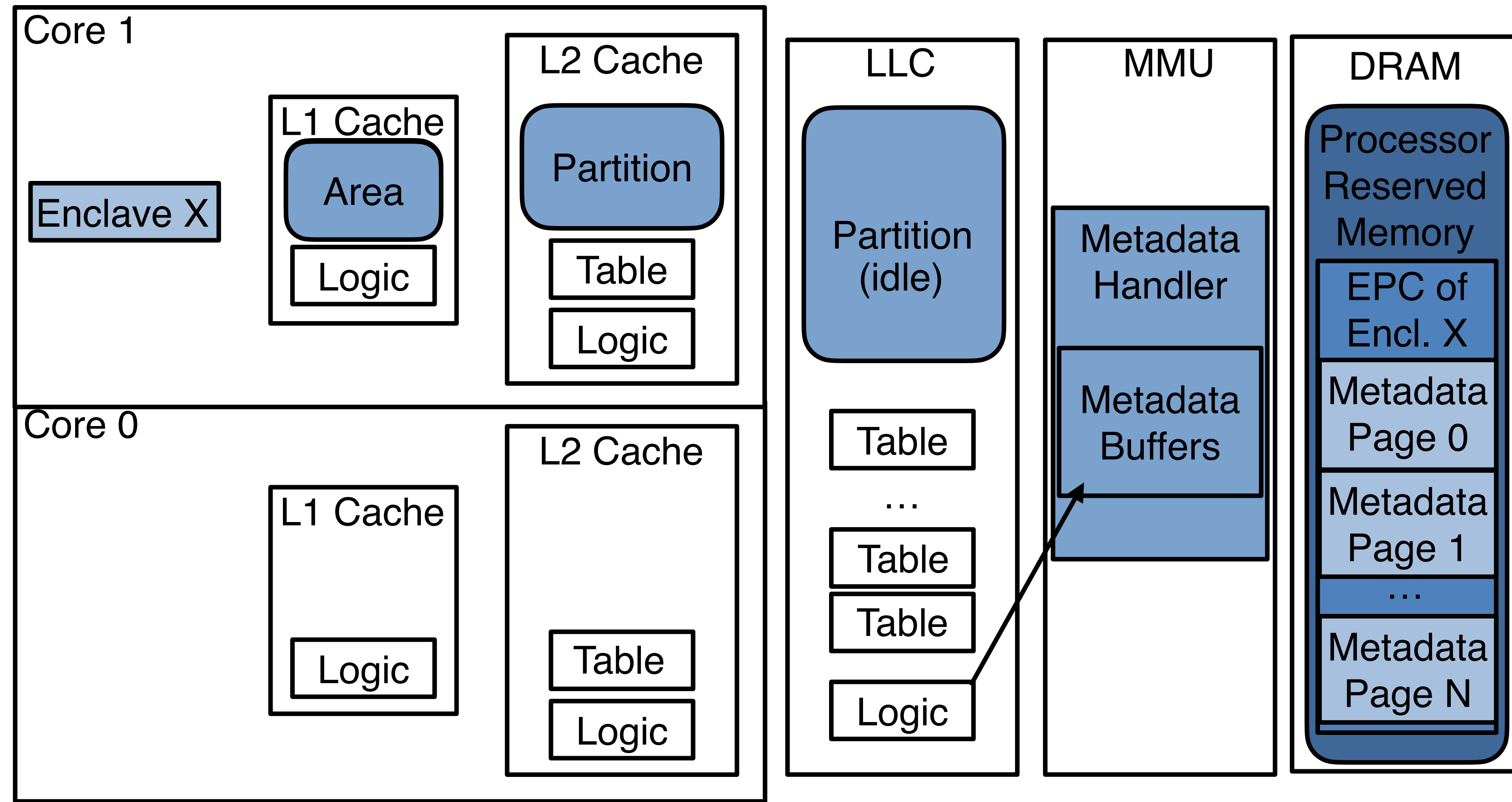
Cache-Aware Context Switches



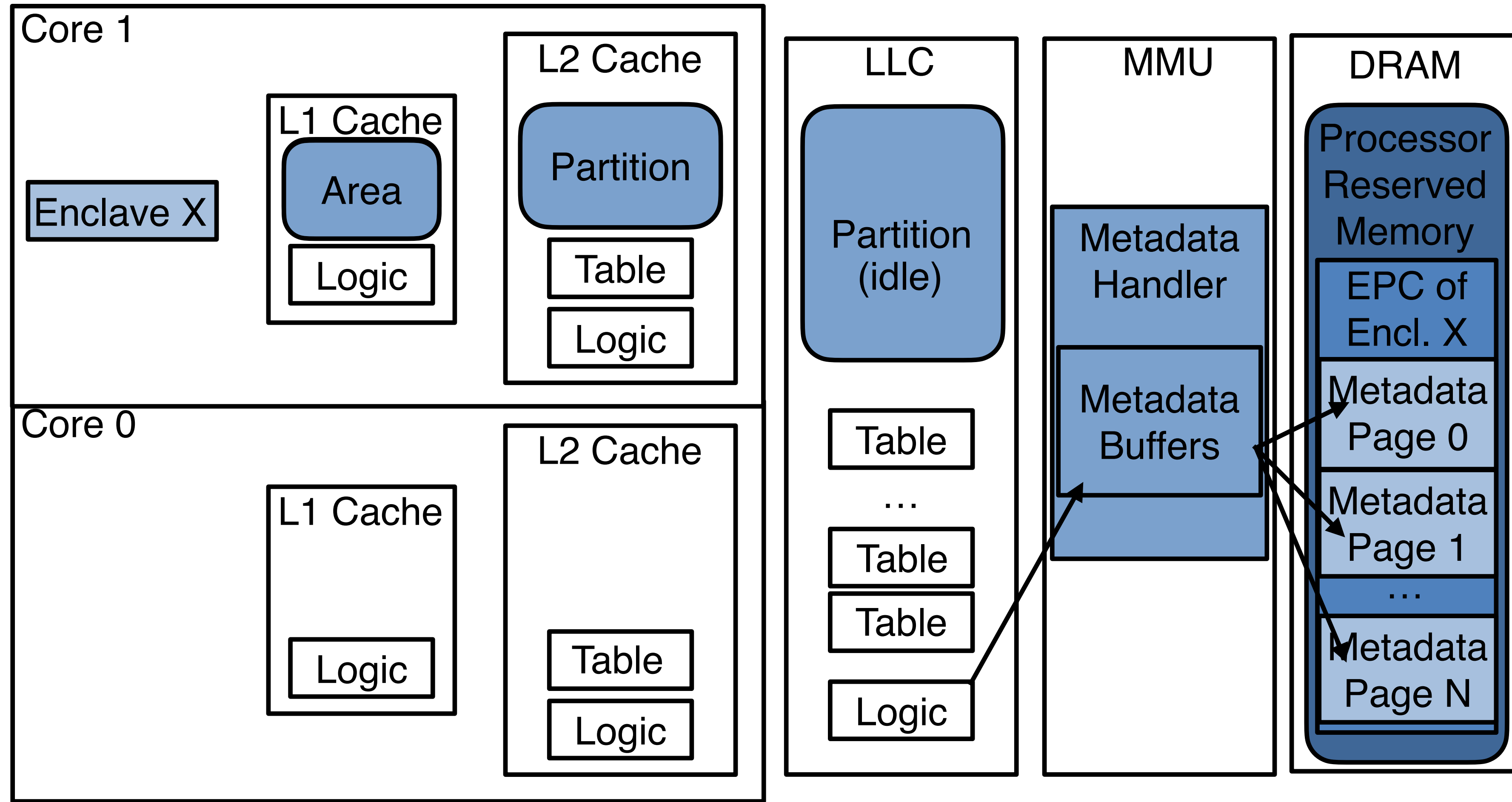
Cache-Aware Context Switches



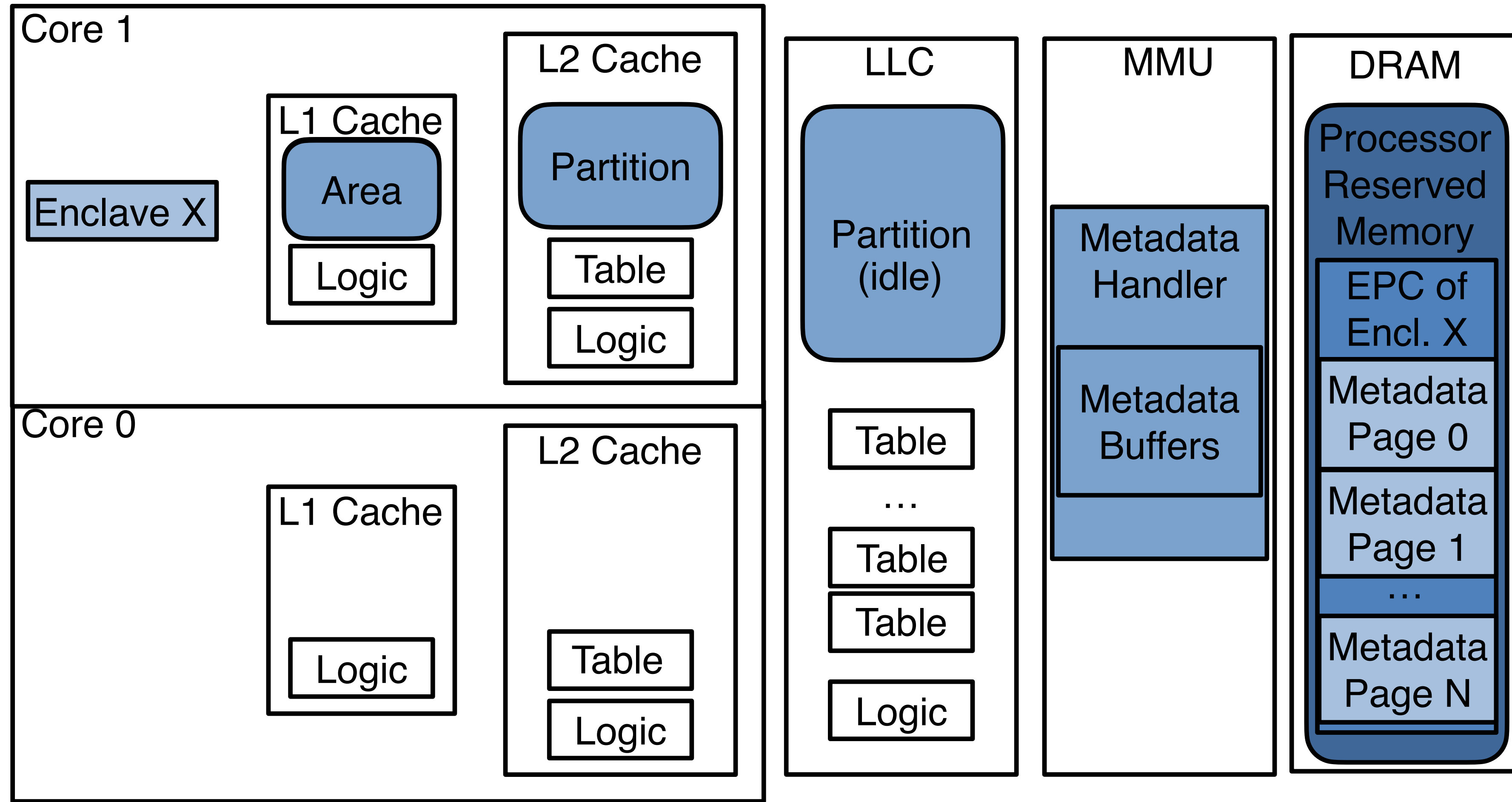
Cache-Aware Context Switches



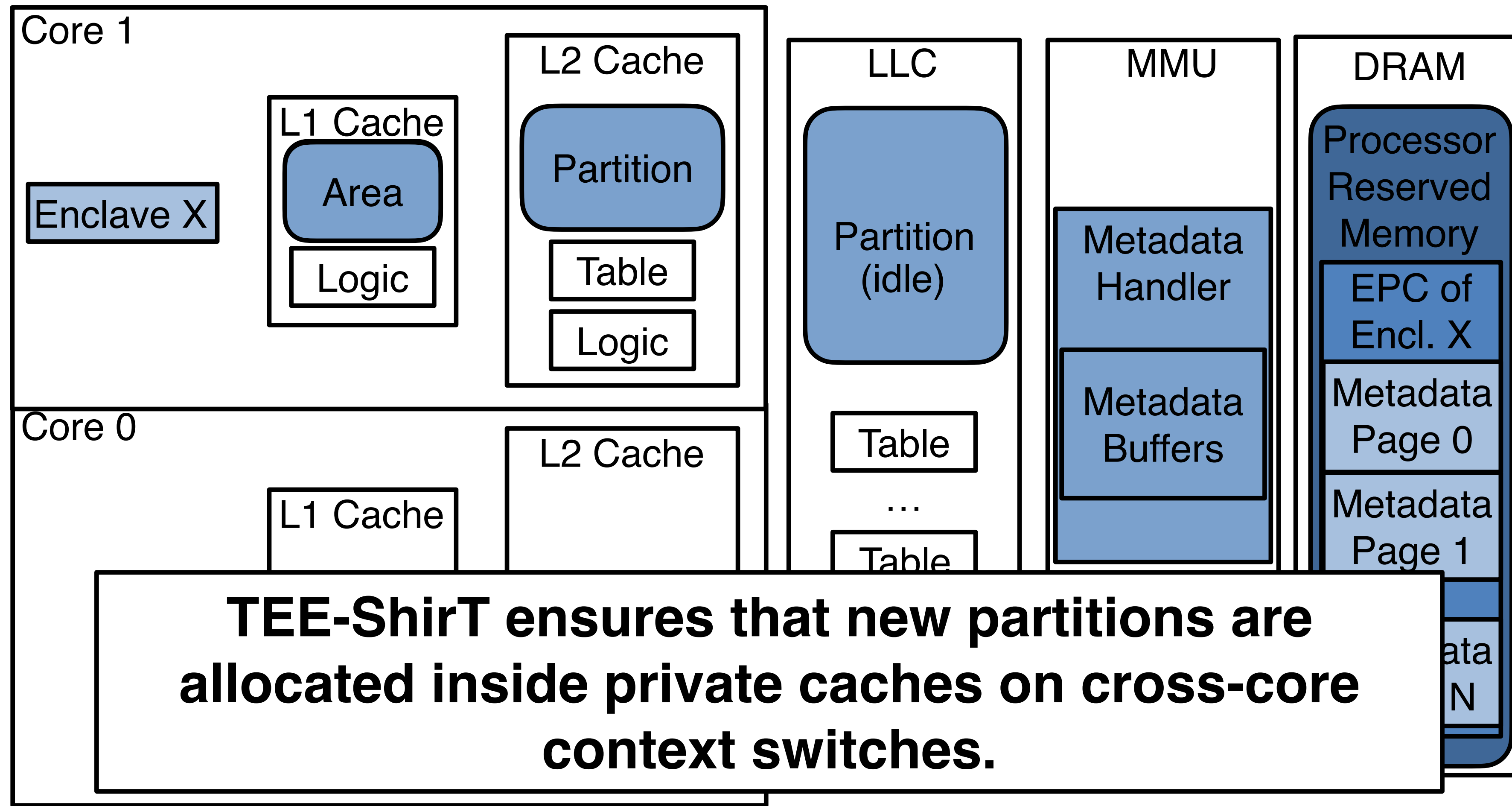
Cache-Aware Context Switches



Cache-Aware Context Switches

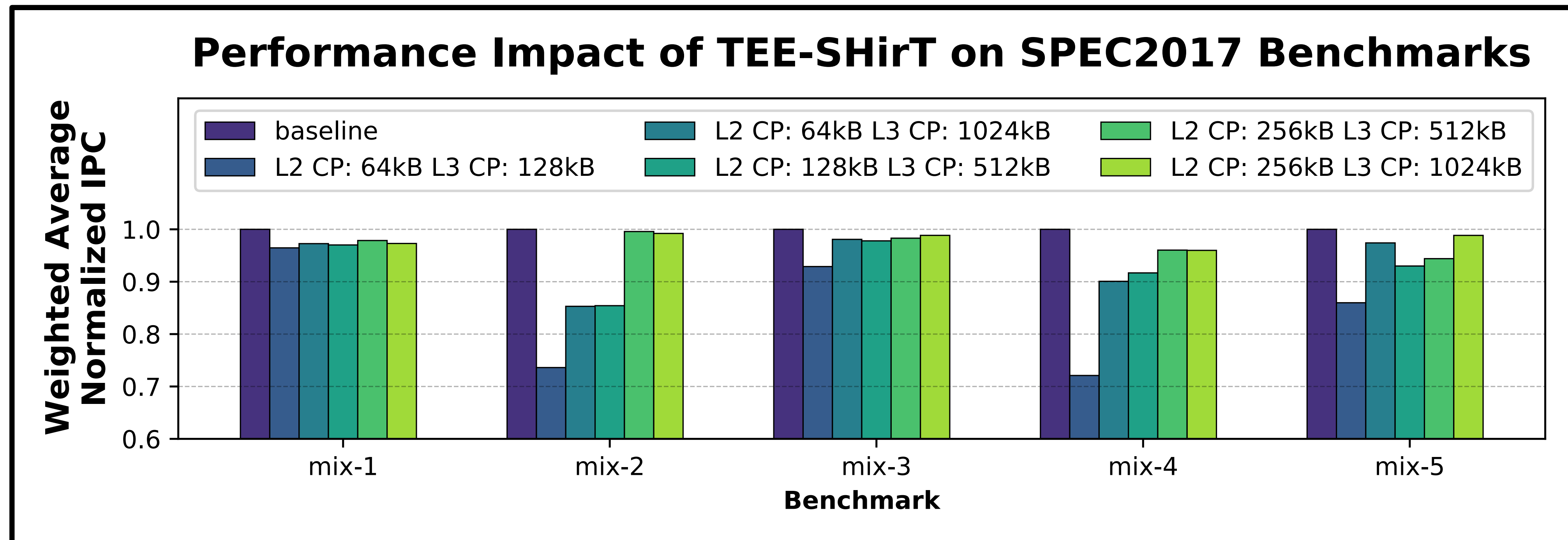


Cache-Aware Context Switches



TEE-ShirT Performance for SPEC2017

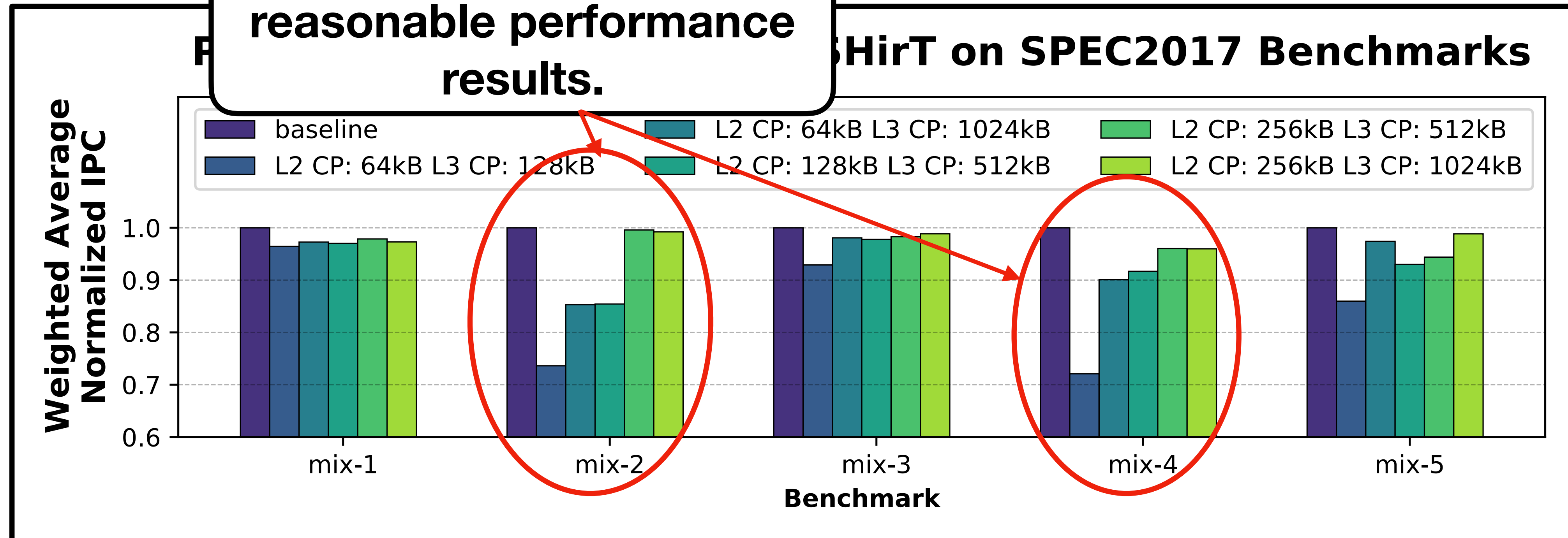
- Evaluation is done on a 4-core system with five benchmarks mixes:



TEE-ShirT Performance for SPEC2017

- Even be... e system with five

With appropriate sizes, even memory-intensive benchmark mixes yield reasonable performance results.



More in the Paper...

TEE-SHirT: Scalable Leakage-Free Cache Hierarchies for TEEs

Kerem Arıkan*, Abraham Farrell*, Williams Zhang Cen*, Jack McMahon*, Barry Williams*, Yu David Liu*, Nael Abu-Ghazaleh†, and Dmitry Ponomarev*

*Binghamton University

†University of California, Riverside

Abstract—Protection of cache hierarchies from side-channel attacks is critical for building secure systems, particularly the ones using Trusted Execution Environments (TEEs). In this paper, we consider the problem of efficient and secure fine-grained partitioning of cache hierarchies and propose a framework, called Secure Hierarchies for TEEs (TEE-SHirT). In the context of a three-level cache system, TEE-SHirT consists of partitioned shared last-level cache, partitioned private L2 caches, and non-partitioned L1 caches that are flushed on context switches and system calls. Efficient and correct partitioning requires careful design. Towards this goal, TEE-SHirT makes three contributions: 1) we demonstrate how the hardware structures used for holding cache partitioning metadata can be effectively virtualized to avoid flushing of cache partition content on context switches and system calls; 2) we show how to support multi-threaded enclaves in TEE-SHirT, addressing the issues of coherency and consistency that arise with both intra-core and inter-core data sharing; 3) we develop a formal security model for TEE-SHirT to rigorously reason about the security of our design.

In this paper, we investigate cache partitioning mechanisms for TEE systems with the goal of protecting the entire hierarchy, and not just a single cache level. Cache partitioning is a principled approach to security that physically isolates applications from each other eliminating leakage due to contention on shared resources. Since cache partitioning is applied to different applications (or enclaves) and they are isolated, the behavior of the victim processes cannot impact any cache-related observations by attackers, making attacks impossible. Existing secure cache partitioning schemes consider only a single level of caches, either private L1 or L2 level caches [23] or shared LLC [41], [44], [56], [63]. These schemes partition caches by ways [23], [41], sets [21], [56], or both [63]. Without loss of generality, we study fine-grained approaches that partition caches by both ways and sets [63].

Various levels of the cache hierarchy require different approaches to achieve security. It has been established that

More in the Paper...

TEE-SHirT: Scalable Leakage-Free Cache Hierarchies for TEEs

Kerem Arıkan*, Abraham Farrell*, Williams Zhang Cen*, Jack McMahon*, Barry Williams*, Yu David Liu*, Nael Abu-Ghazaleh†, and Dmitry Ponomarev*

*Binghamton University

†University of California, Riverside

Abstract—Protection of cache hierarchies from side-channel attacks is critical for building secure systems, particularly the ones using Trusted Execution Environments (TEEs). In this paper, we consider the problem of efficient and secure fine-grained partitioning of cache hierarchies and propose a framework, called Secure Hierarchies for TEEs (TEE-SHirT). In the context of a three-level cache system, TEE-SHirT consists of partitioned shared last-level cache, partitioned private L2 caches, and non-partitioned L1 caches that are flushed on context switches and system calls. Efficient and correct partitioning requires careful design. Towards this goal, TEE-SHirT makes three contributions: 1) we demonstrate how the hardware structures used for holding cache partitioning metadata can be effectively virtualized to avoid flushing of cache partition content on context switches and system calls; 2) we show how to support multi-threaded enclaves in TEE-SHirT, addressing the issues of coherency and consistency that arise with both intra-core and inter-core data sharing; 3) we develop a formal security model for TEE-SHirT to rigorously reason about the security of our design.

In this paper, we investigate cache partitioning mechanisms for TEE systems with the goal of protecting the entire hierarchy, and not just a single cache level. Cache partitioning is a principled approach to security that physically isolates applications from each other eliminating leakage due to contention on shared resources. Since cache partitioning is applied to different applications (or enclaves) and they are isolated, the behavior of the victim processes cannot impact any cache-related observations by attackers, making attacks impossible. Existing secure cache partitioning schemes consider only a single level of caches, either private L1 or L2 level caches [23] or shared LLC [41], [44], [56], [63]. These schemes partition caches by ways [23], [41], sets [21], [56], or both [63]. Without loss of generality, we study fine-grained approaches that partition caches by both ways and sets [63].

Various levels of the cache hierarchy require different approaches to achieve security. It has been established that

- Formal security analysis

More in the Paper...

TEE-SHirT: Scalable Leakage-Free Cache Hierarchies for TEEs

Kerem Arıkan*, Abraham Farrell*, Williams Zhang Cen*, Jack McMahon*, Barry Williams*, Yu David Liu*, Nael Abu-Ghazaleh†, and Dmitry Ponomarev*

*Binghamton University

†University of California, Riverside

Abstract—Protection of cache hierarchies from side-channel attacks is critical for building secure systems, particularly the ones using Trusted Execution Environments (TEEs). In this paper, we consider the problem of efficient and secure fine-grained partitioning of cache hierarchies and propose a framework, called Secure Hierarchies for TEEs (TEE-SHirT). In the context of a three-level cache system, TEE-SHirT consists of partitioned shared last-level cache, partitioned private L2 caches, and non-partitioned L1 caches that are flushed on context switches and system calls. Efficient and correct partitioning requires careful design. Towards this goal, TEE-SHirT makes three contributions: 1) we demonstrate how the hardware structures used for holding cache partitioning metadata can be effectively virtualized to avoid flushing of cache partition content on context switches and system calls; 2) we show how to support multi-threaded enclaves in TEE-SHirT, addressing the issues of coherency and consistency that arise with both intra-core and inter-core data sharing; 3) we develop a formal security model for TEE-SHirT to rigorously reason about the security of our design.

In this paper, we investigate cache partitioning mechanisms for TEE systems with the goal of protecting the entire hierarchy, and not just a single cache level. Cache partitioning is a principled approach to security that physically isolates applications from each other eliminating leakage of information on shared resources. Since cache partitioning is applied to different applications (or enclaves) and they are isolated, the behavior of the victim processes cannot impact any cache-related observations by attackers, making attacks impossible. Existing secure cache partitioning schemes consider only a single level of caches, either private L1 or L2 level caches [23] or shared LLC [41], [44], [56], [63]. These schemes partition caches by ways [23], [41], sets [21], [56], or both [63]. Without loss of generality, we study fine-grained approaches that partition caches by both ways and sets [63].

Various levels of the cache hierarchy require different approaches to achieve security. It has been established that

- Formal security analysis
- Cache coherence support

More in the Paper...

TEE-SHirT: Scalable Leakage-Free Cache Hierarchies for TEEs

Kerem Arıkan*, Abraham Farrell*, Williams Zhang Cen*, Jack McMahon*, Barry Williams*, Yu David Liu*, Nael Abu-Ghazaleh†, and Dmitry Ponomarev*

*Binghamton University

†University of California, Riverside

Abstract—Protection of cache hierarchies from side-channel attacks is critical for building secure systems, particularly the ones using Trusted Execution Environments (TEEs). In this paper, we consider the problem of efficient and secure fine-grained partitioning of cache hierarchies and propose a framework, called Secure Hierarchies for TEEs (TEE-SHirT). In the context of a three-level cache system, TEE-SHirT consists of partitioned shared last-level cache, partitioned private L2 caches, and non-partitioned L1 caches that are flushed on context switches and system calls. Efficient and correct partitioning requires careful design. Towards this goal, TEE-SHirT makes three contributions: 1) we demonstrate how the hardware structures used for holding cache partitioning metadata can be effectively virtualized to avoid flushing of cache partition content on context switches and system calls; 2) we show how to support multi-threaded enclaves in TEE-SHirT, addressing the issues of coherency and consistency that arise with both intra-core and inter-core data sharing; 3) we develop a formal security model for TEE-SHirT to rigorously reason about the security of our design.

In this paper, we investigate cache partitioning mechanisms for TEE systems with the goal of protecting the entire hierarchy, and not just a single cache level. Cache partitioning is a principled approach to security that physically isolates applications from each other eliminating leakage of information on shared resources. Since cache partitioning is applied to different applications (or enclaves) and each application is isolated, the behavior of the victim process cannot impact any cache-related observations by attackers, making attacks impossible. Existing secure cache partitioning schemes consider only a single level of caches, either private L1 or L2 level caches [23] or shared LLC [41], [44], [56], [63]. These schemes partition caches by ways [23], [41], sets [21], [56], or both [63]. Without loss of generality, we study fine-grained approaches that partition caches by both ways and sets [63]. Various levels of the cache hierarchy require different approaches to achieve security. It has been established that

- Formal security analysis
- Cache coherence support
- Integration with Intel SGX

Questions