UTOPIA

Fast and Efficient Address Translation via Hybrid Restrictive & Flexible Virtual-to-Physical Address Mappings

Konstantinos Kanellopoulos
Rahul Bera, Kosta Stojiljkovic, Nisa Bostanci, Can Firtina,
Rachata Ausavarungnirun, Rakesh Kumar, Nastaran Hajinazar,
Mohammad Sadrosadati, Nandita Vijaykumar, and Onur Mutlu

Executive Summary

Problem: Conventional virtual memory (VM) frameworks enable a virtual address to flexibly map to any physical address. This flexibility necessitates large translation structures leading to:

(1) high translation **latency** and (2) large translation-induced **interference** in the memory hierarchy

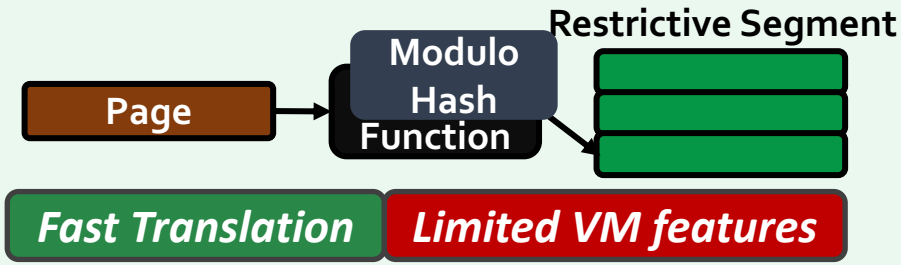
Motivation: Restricting the address mapping leads to compact translation structures and reduces the overheads of address translation. Doing so across the **entire memory** has two major drawbacks:

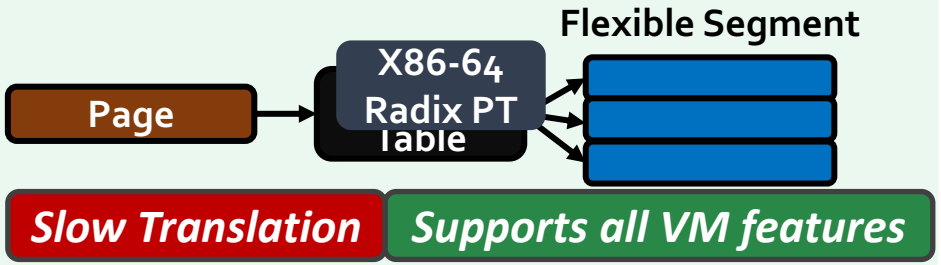
(1) Limits core VM functionalities (e.g., data sharing)

(2) Increases swapping activity in the presence of free physical memory

Key Idea: Utopia is a new hybrid virtual-to-physical address mapping scheme that allows both **flexible** and **restrictive** hash-based address mappings to **harmoniously co-exist** in the system

Utopia manages physical memory using two types of physical memory segments:





Key Results: Outperforms (i) the state-of-the-art **contiguity-aware translation scheme by 13%,** and (ii) achieves **95% of the performance of an ideal perfect-TLB**

https://github.com/CMU-SAFARI/Utopia

Talk Outline

Virtual Memory Background

Address Translation Overheads

Utopia: Hybrid Address Mappings

Utopia: Key Challenges

Evaluation Results

Talk Outline

Virtual Memory Background

Address Translation Overheads

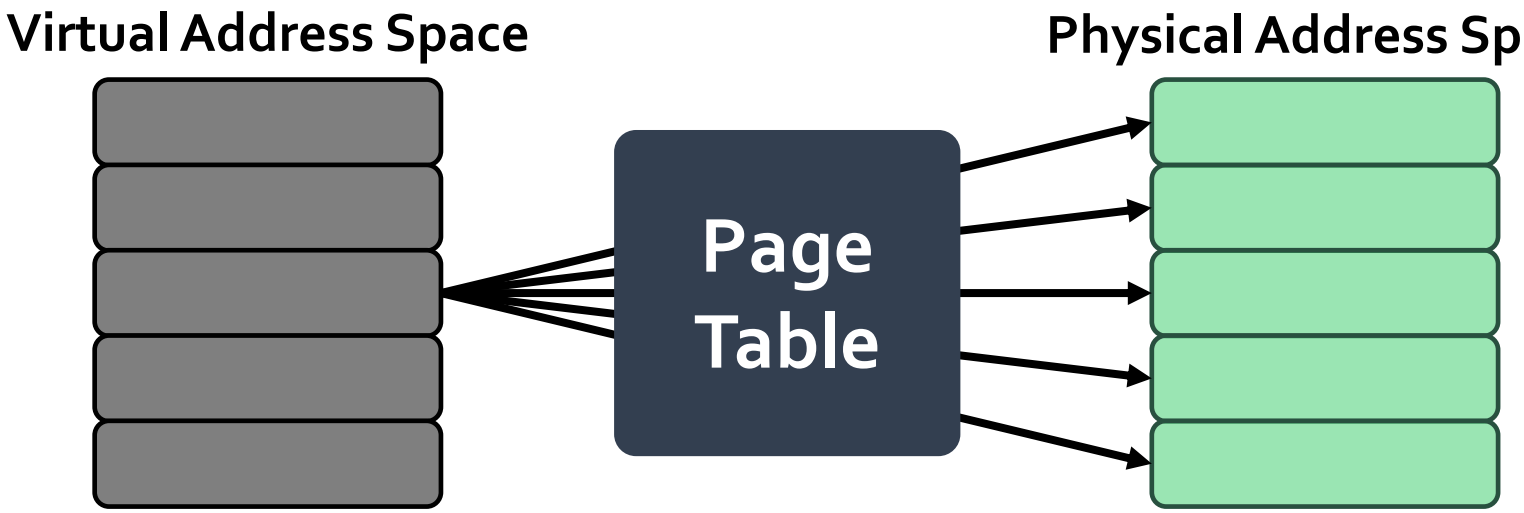
Utopia: Hybrid Address Mappings

Utopia: Key Challenges

Evaluation Results

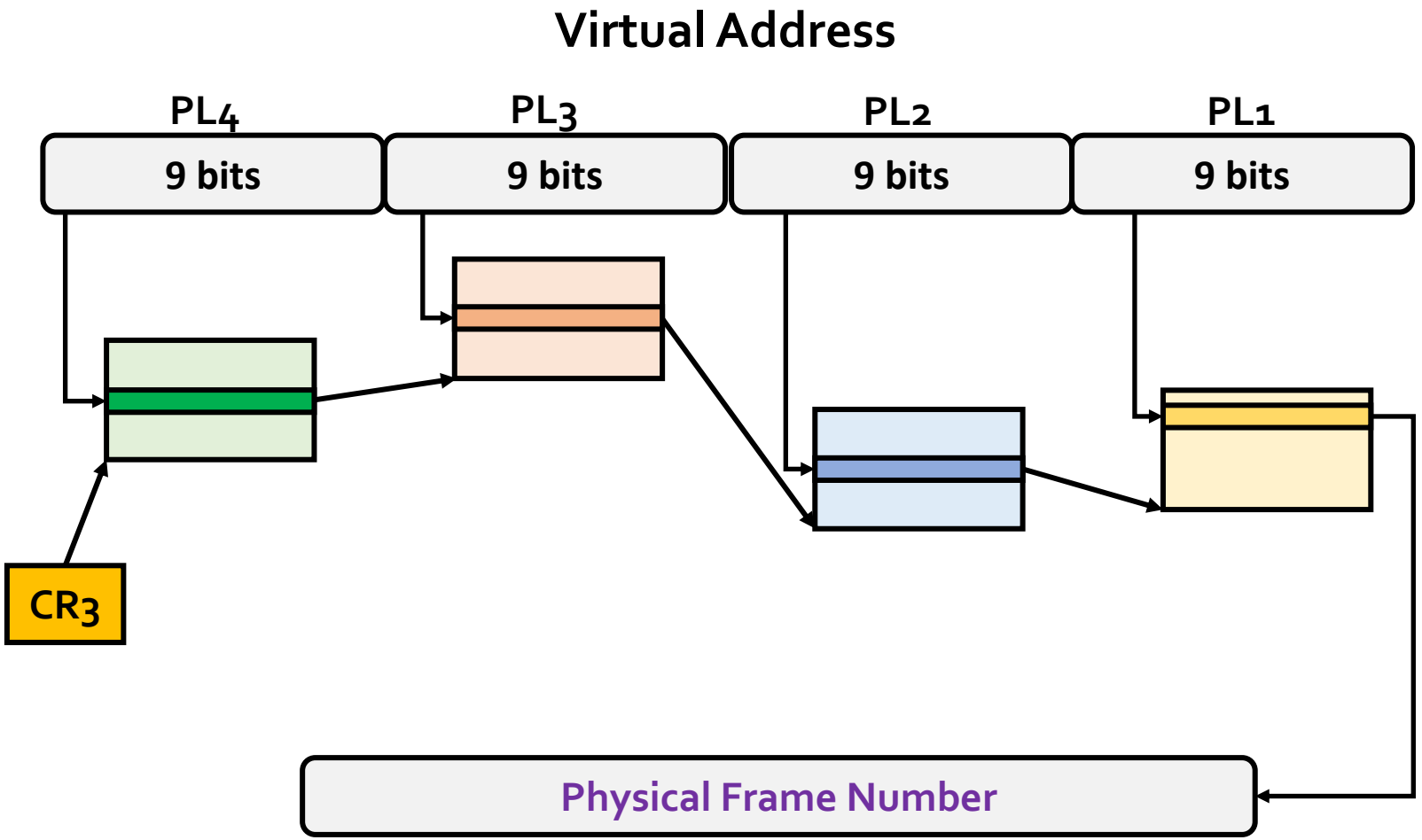
Virtual Memory Basics

A core feature of virtual memory management is that the **mapping** between virtual-to-physical pages is **fully-associative**

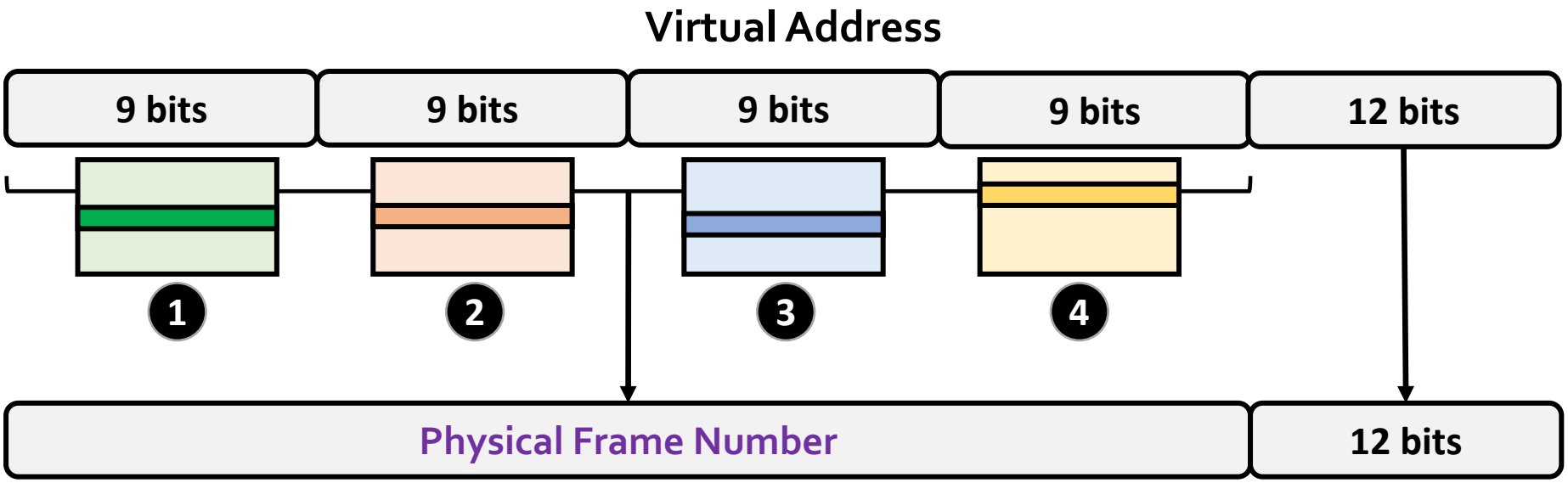


Perform **Page Table Walk (PTW)** to retrieve the mapping

Page Table Walk in x86-64

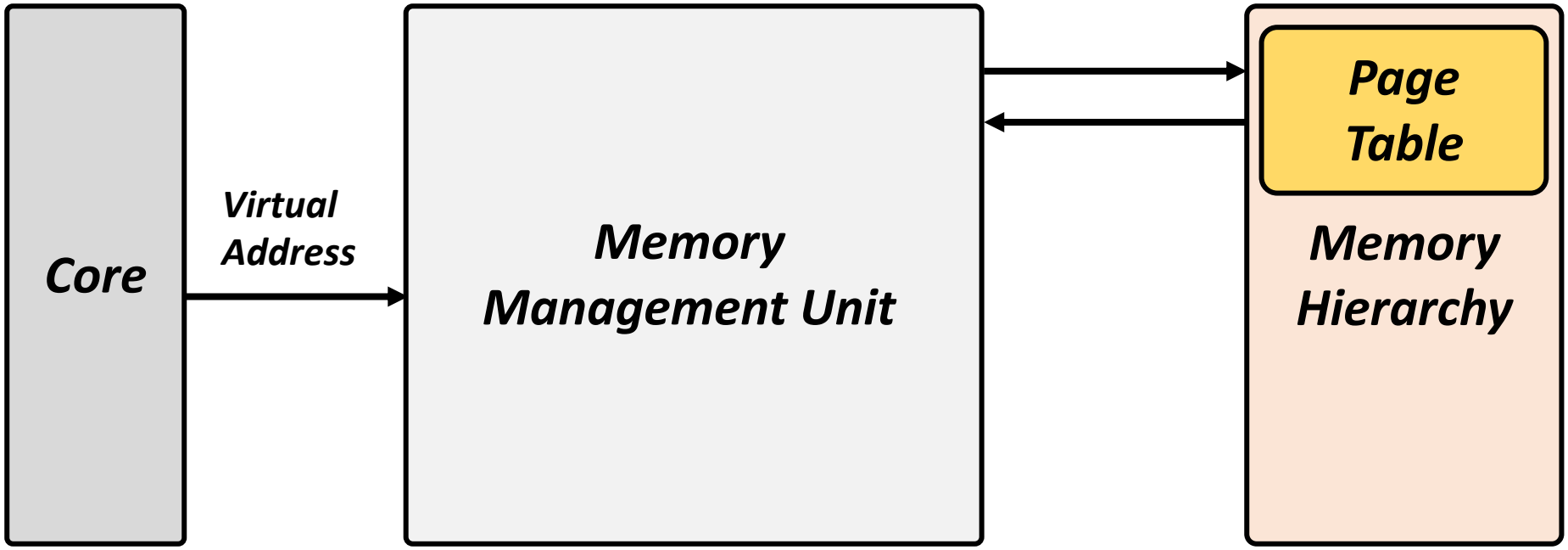


Page Table Walk in x86-64

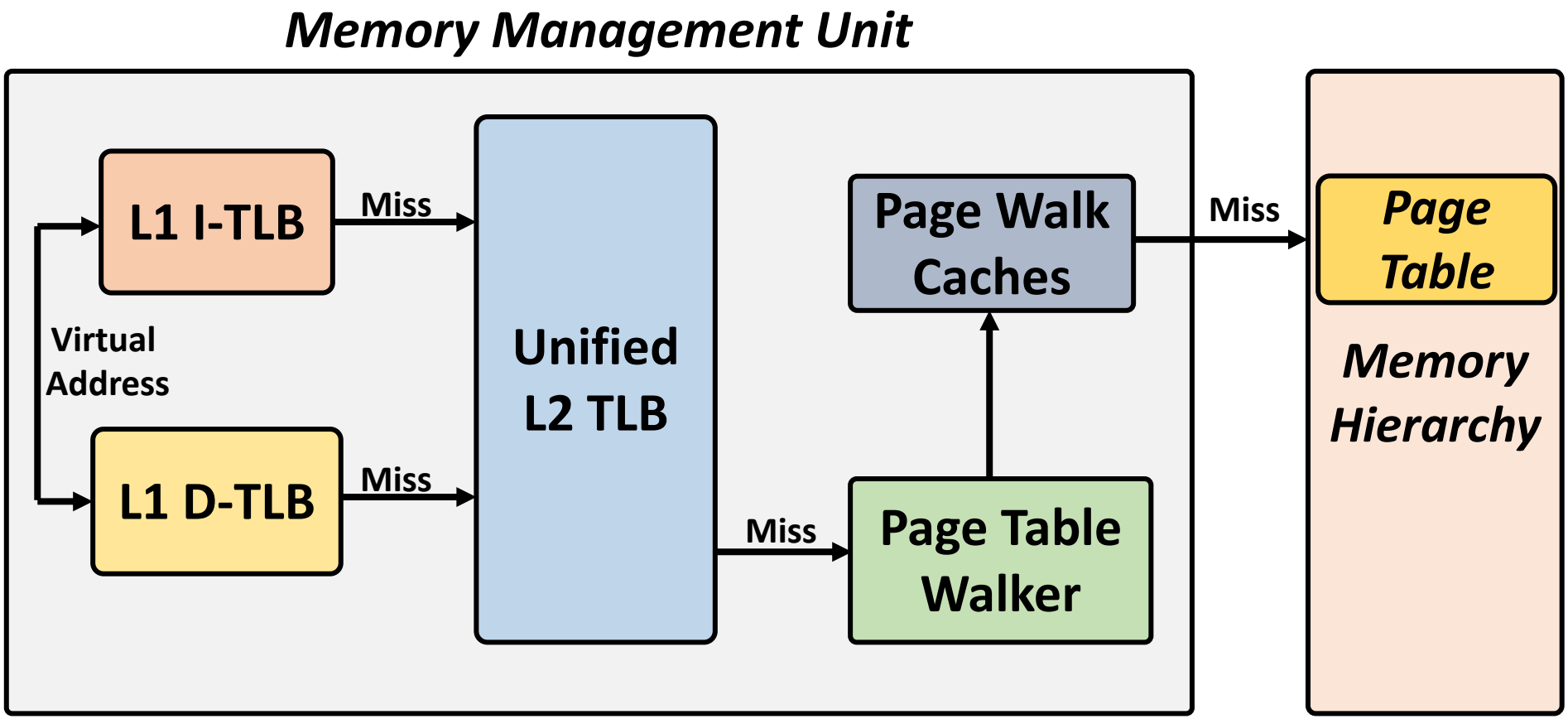


***Four* sequential memory accesses during a page table walk in x86-64**

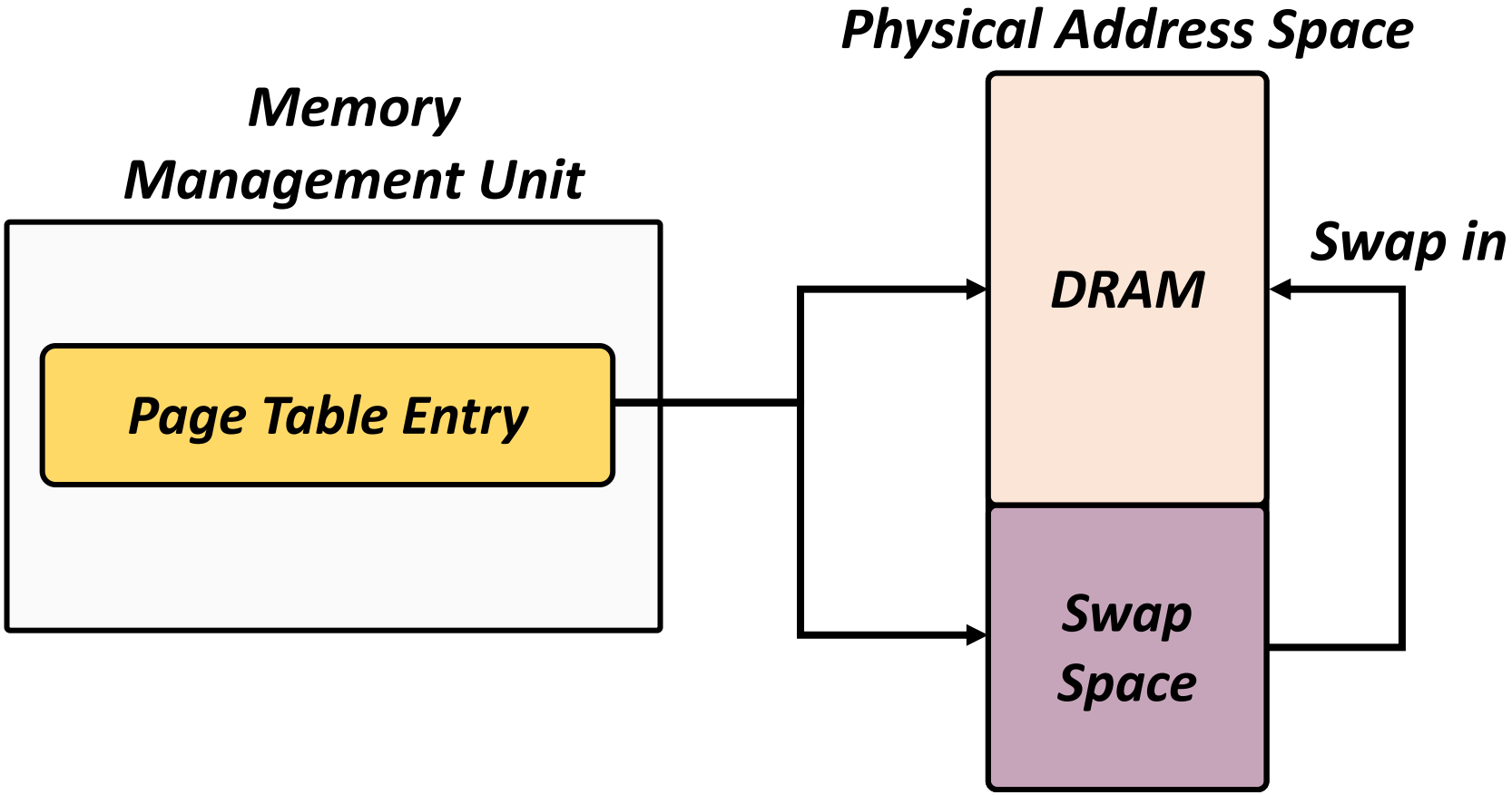
Address Translation Flow (I)



Address Translation Flow (II)



Address Translation Flow (III)



Talk Outline

Virtual Memory Background

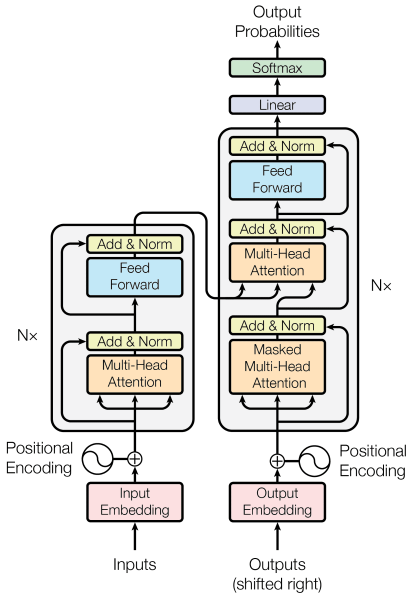
Address Translation Overheads

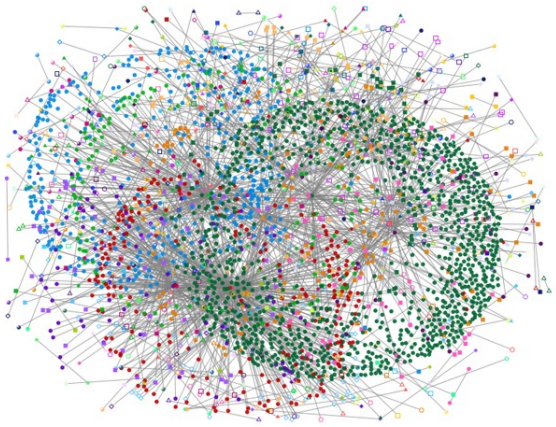
Utopia: Hybrid Address Mappings

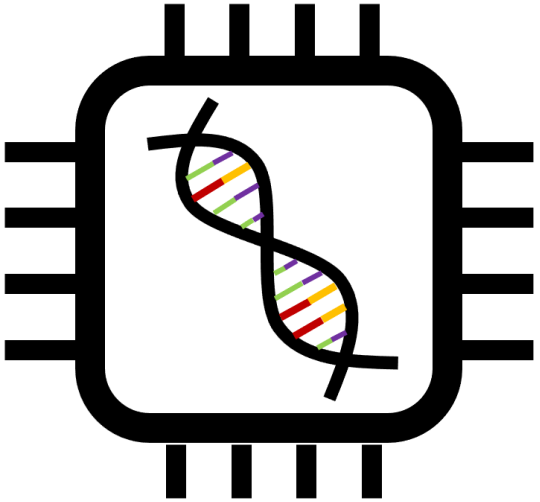
Utopia: Key Challenges

Evaluation Results

Data-Intensive Workloads







Generative AI | **Graph Analytics** | **Bioinformatics**

High address translation overheads

Address Translation Overhead

High-latency page table walks

Frequent page table walks

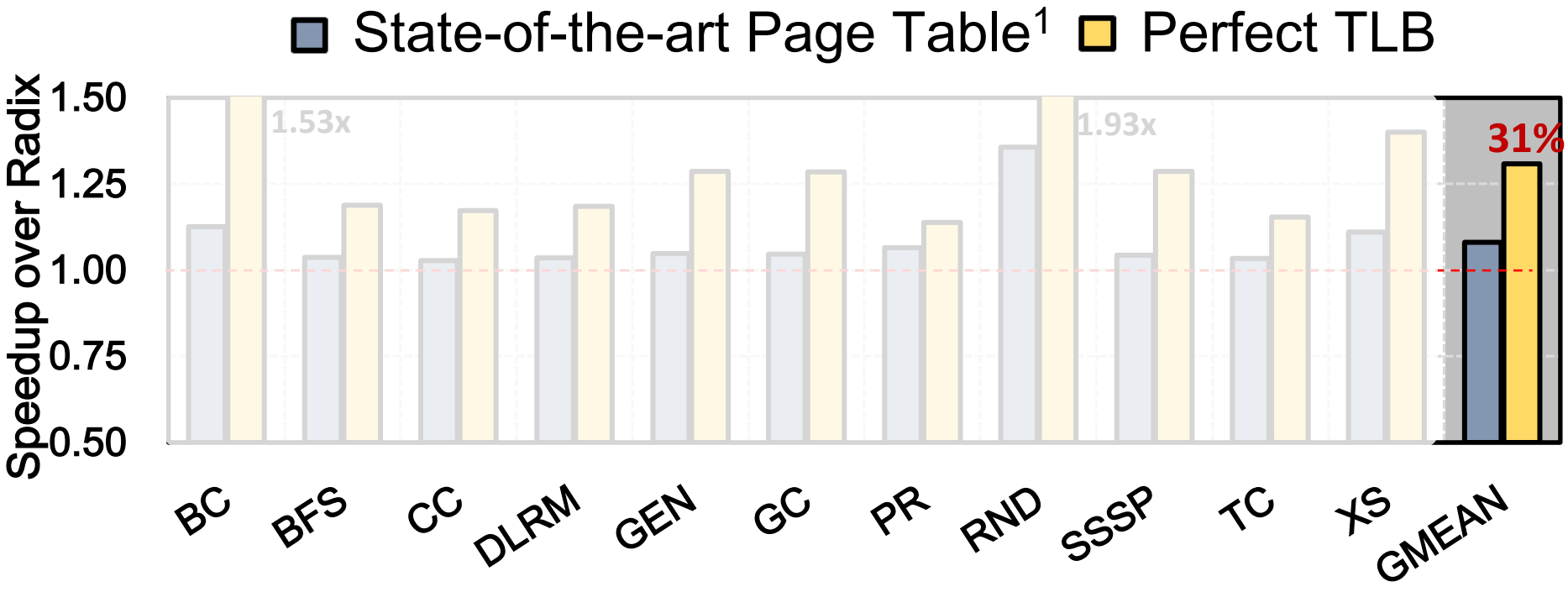
Address Translation Overhead

High latency ***PTWs*** + ***Frequent*** ***PTWs***

High performance overheads

High interference in memory hierarchy

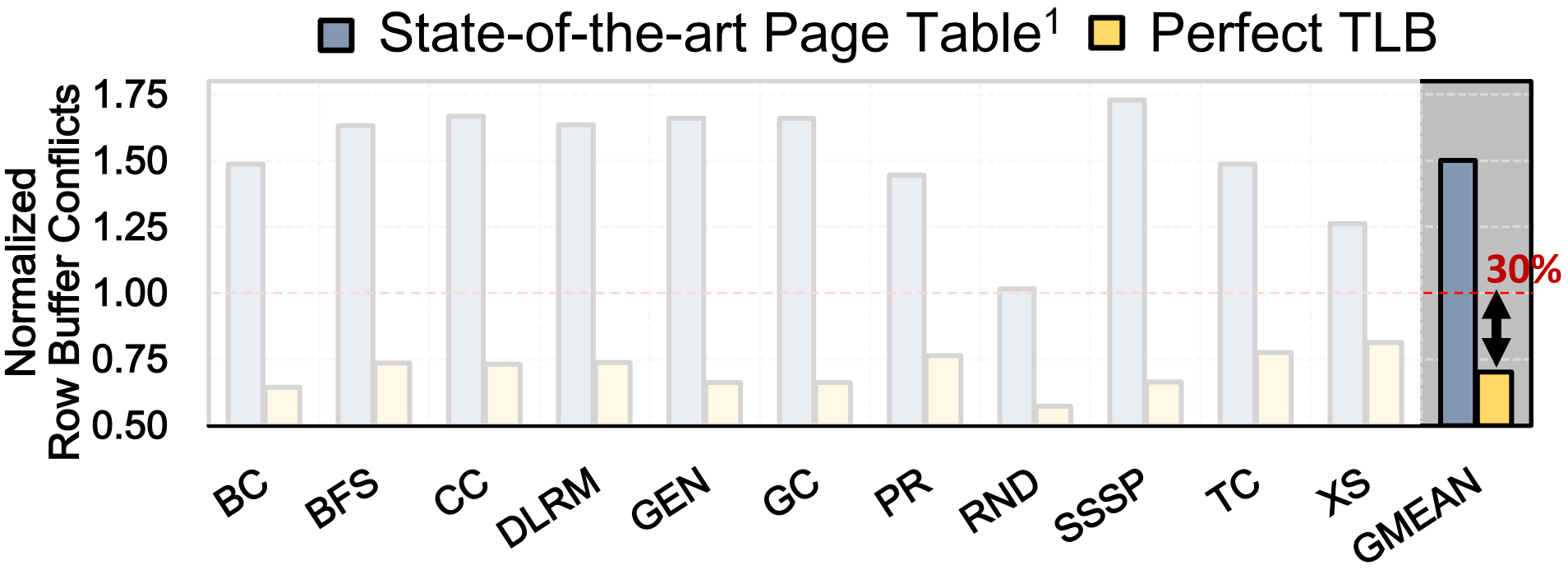
High Performance Overhead



Completely avoiding address translation leads on average to 31% higher performance

[1] Skarlatos et al. "Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism" ASPLOS 2020

High Interference in Main Memory

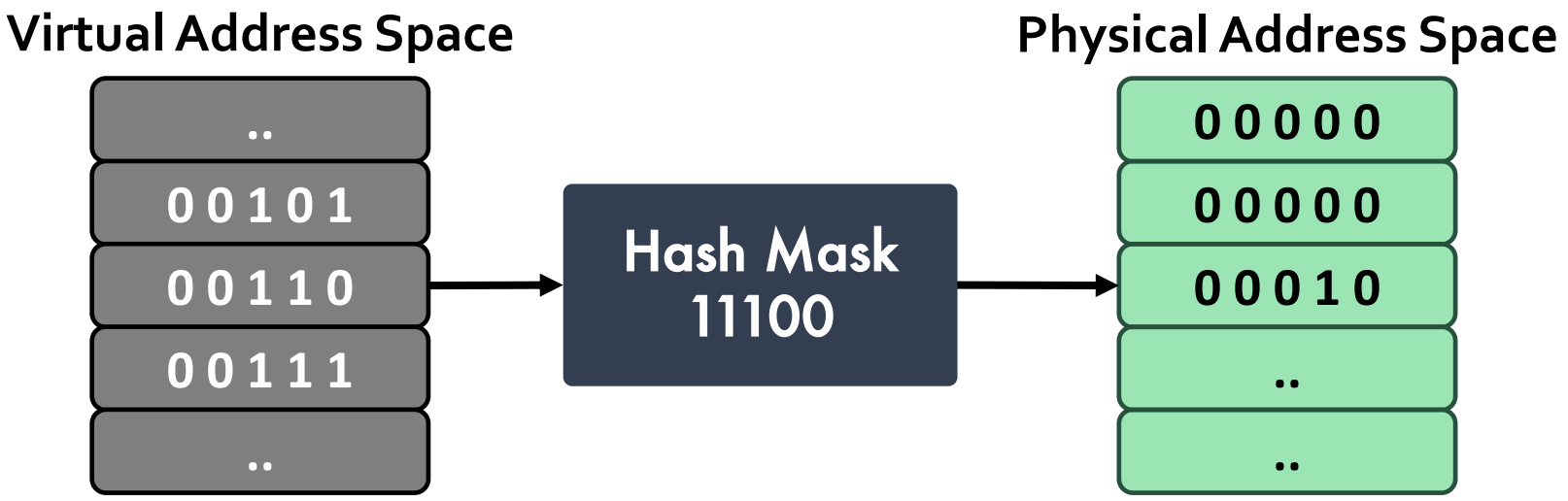


Completely avoiding address translation leads to 30% fewer DRAM row buffer conflicts

[1] Skarlatos et al. "Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism" ASPLOS 2020

Idea: Restricting VA-to-PA Mapping

Restrict the VA-to-PA mapping to perform fast address translation[1,2]

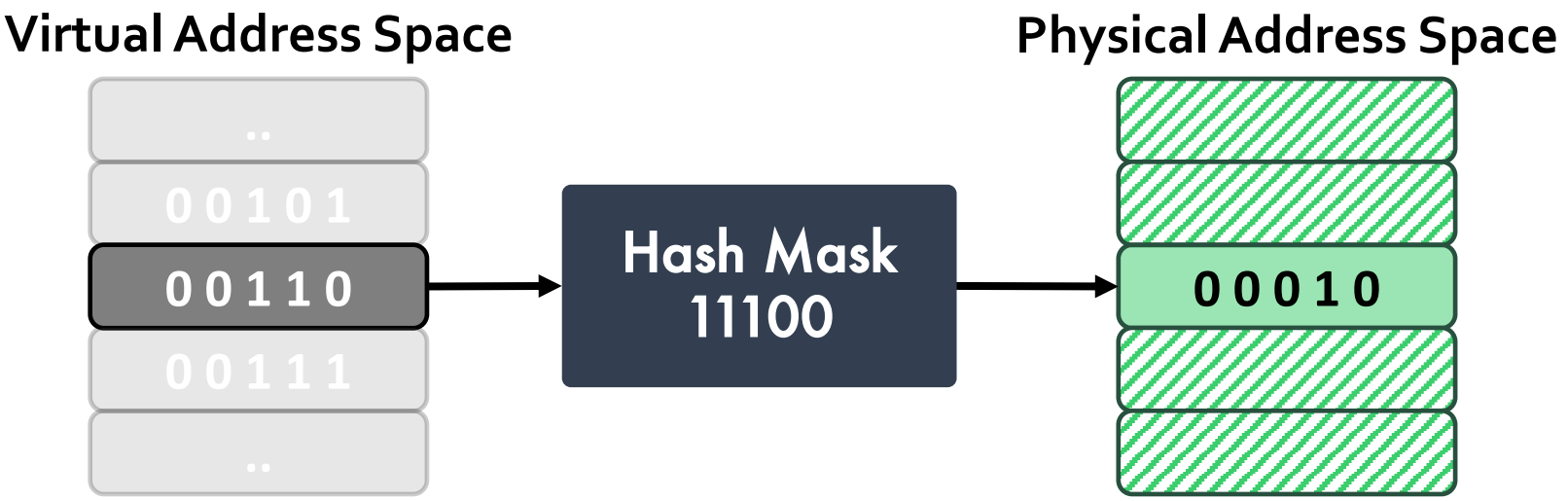


[1] Picorel et al. "Near-Memory Address Translation" PACT 2017

[2] Gosakan et al. "Mosaic Pages: Big TLB Reach with Small Pages" ASPLOS 2023

Idea: Restricting VA-to-PA Mapping

Restrict the VA-to-PA mapping to perform fast address translation[1,2]



[1] Picorel et al. "Near-Memory Address Translation" PACT 2016

[2] Gosakan et al. "Mosaic Pages: Big TLB Reach with Small Pages" ASPLOS 2023

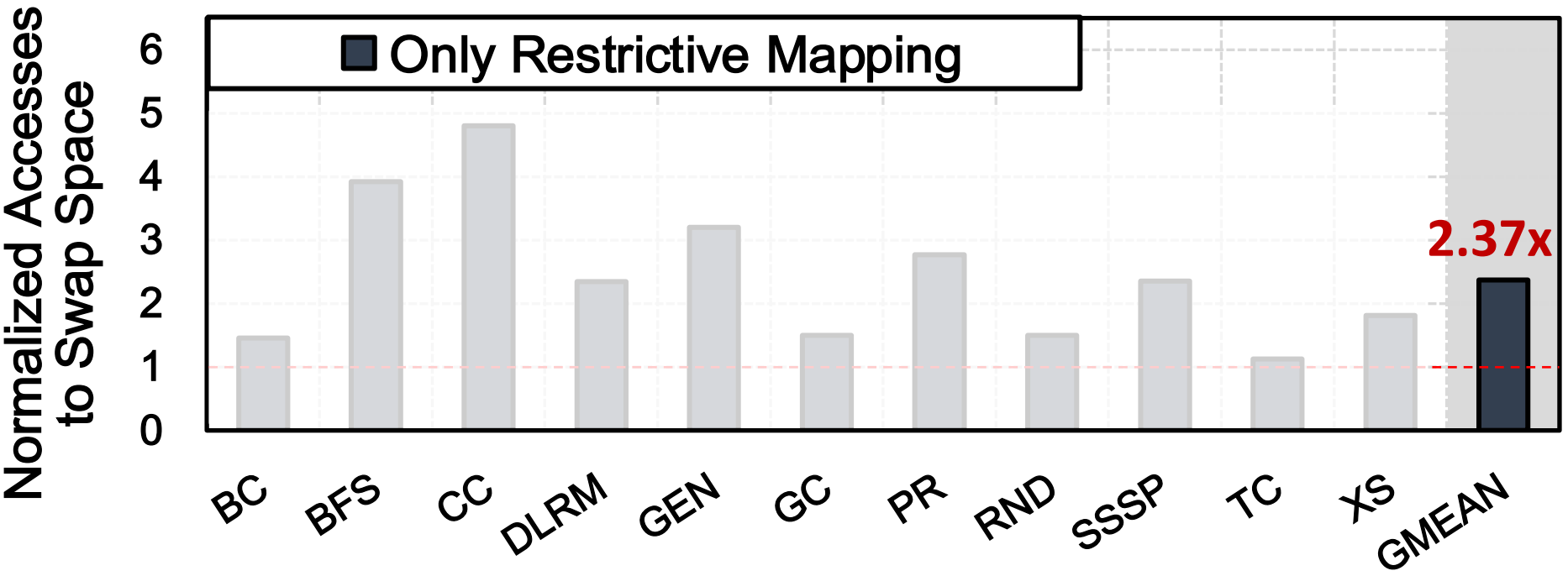
Drawbacks of Restricting VA-to-PA Mapping

Employing a restrictive mapping across the entire memory comes with two key drawbacks:

1. Limits core **VM functionalities** such as data sharing

2. Increases **swapping activity** since the system cannot map virtual pages to free physical pages

Effect on Swapping Activity



Sole use of restrictive mapping leads to 2.37x higher swapping activity over the baseline

Our Goal

Design a virtual-to-physical address mapping scheme that:

- Provides fast and efficient translation using a **restrictive hash-based** address mapping
- Enjoys the benefits of the conventional **fully-flexible** address mapping

Talk Outline

Virtual Memory Background

Address Translation Overheads

Utopia: Hybrid Address Mappings

Utopia: Key Challenges

Evaluation Results

Utopia: Key Idea

We propose **Utopia**, a new virtual-to-physical mapping scheme that enables both:

Restrictive Mapping **Flexible Mapping**

Harmoniously co-exist in the system

Utopia: Key Idea

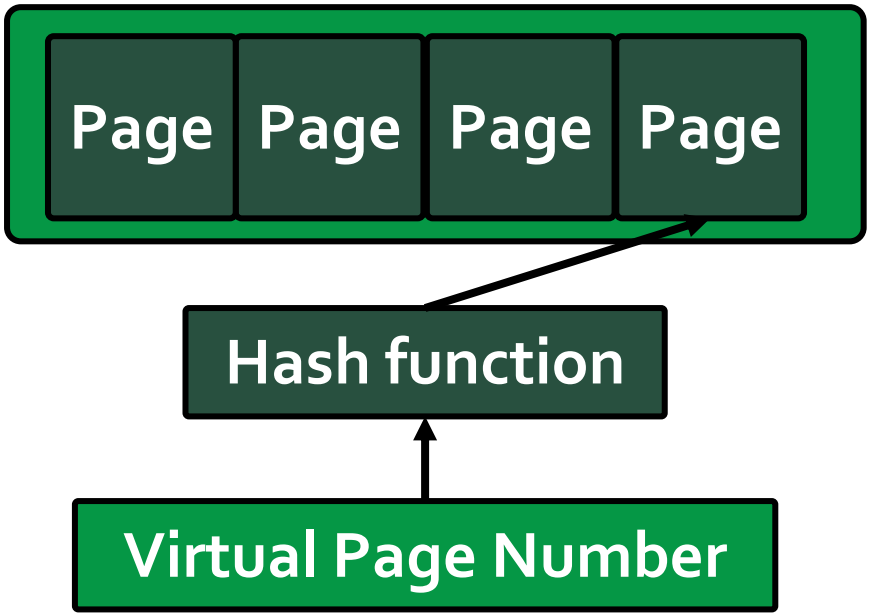
Manage physical memory using two types of physical memory segments:

Restrictive Segments

Flexible Segments

Utopia: Key Idea (I)

Restrictive Segment (RestSeg)



Fast address translation

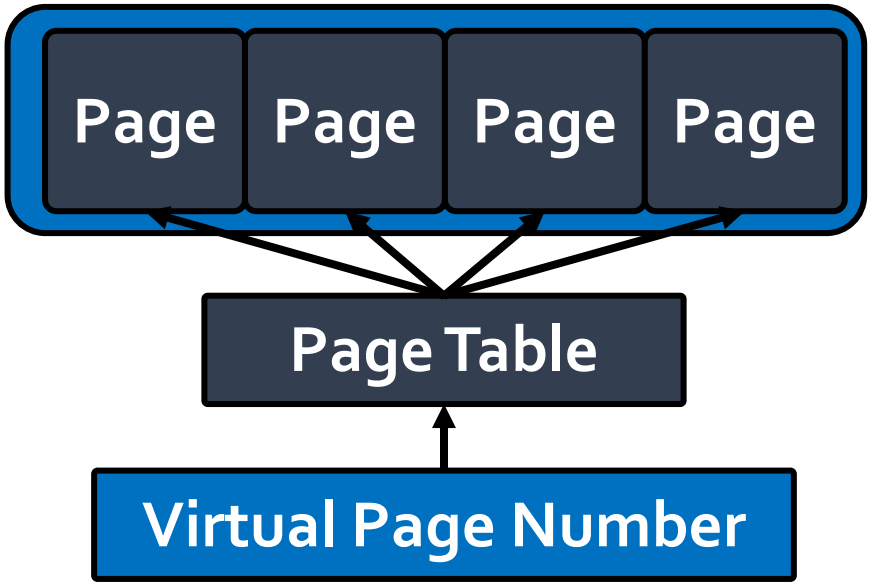
Limited VM functionalities

Flexible Segments

Utopia: Key Idea (II)

Restrictive Segments

Flexible Segment (FlexSeg)



Supports all conventional VM features

High-latency address translation

RestSeg Properties

Structural Properties

Address Translation for Data in RestSeg

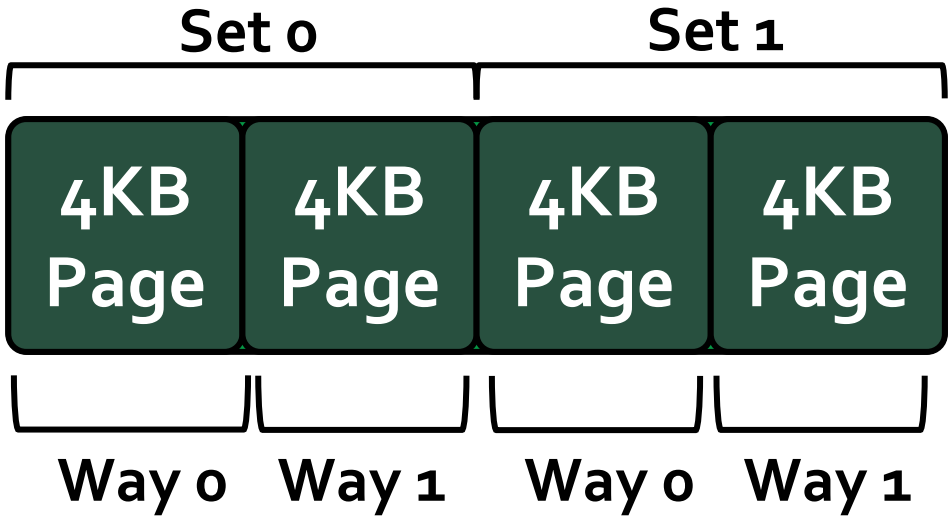
RestSeg Properties

Structural Properties

RestSeg is organized in
a **set-associative** manner
similar to how hardware caches operate

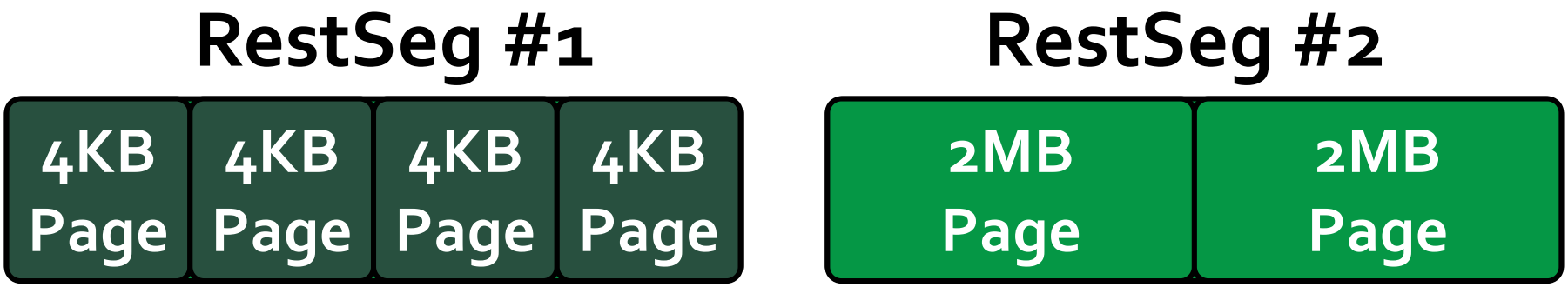
RestSeg: Structural Properties

Example: 2-way associative RestSeg with 2 sets



Set-associative design offers high flexibility

Multiple RestSegs in the System



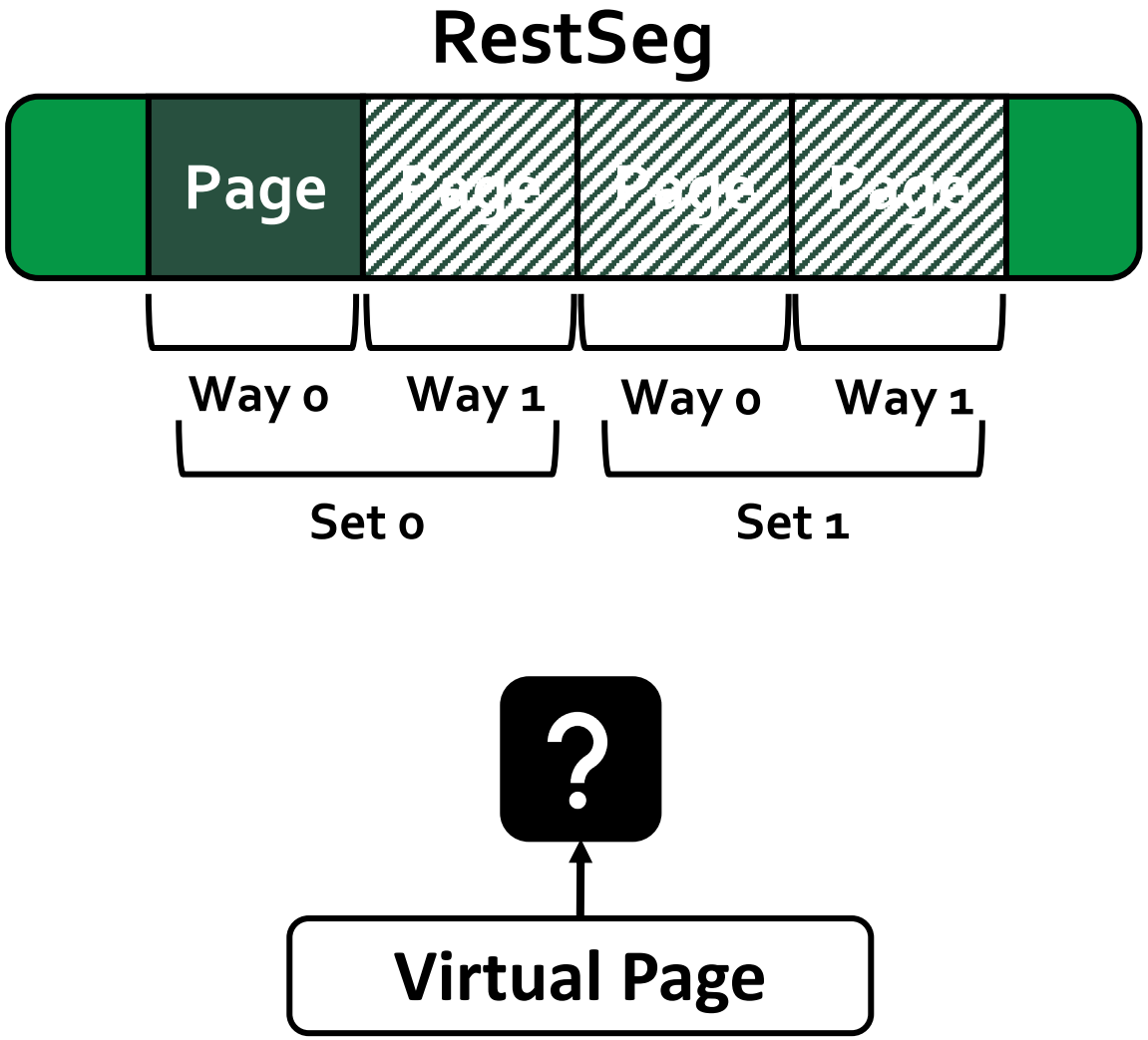
Backward compatible with large page mechanisms

RestSeg Properties

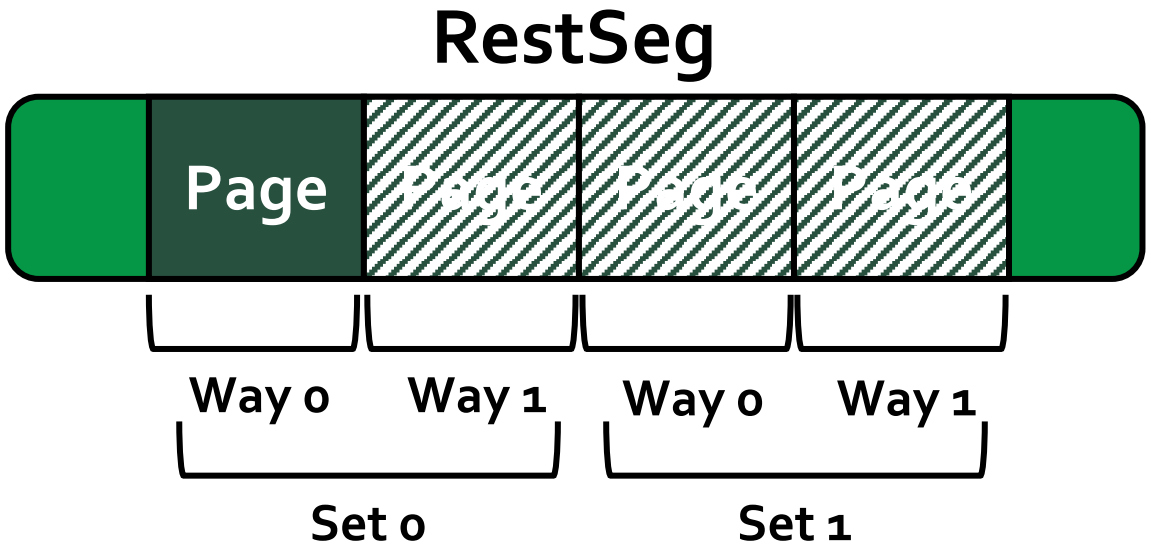
Structural Properties

Address Translation for Data in RestSeg

Restrictive Segment Walk (RSW)



Restrictive Segment Walk (RSW)



How can we find out the physical location of the virtual page?

Virtual Page

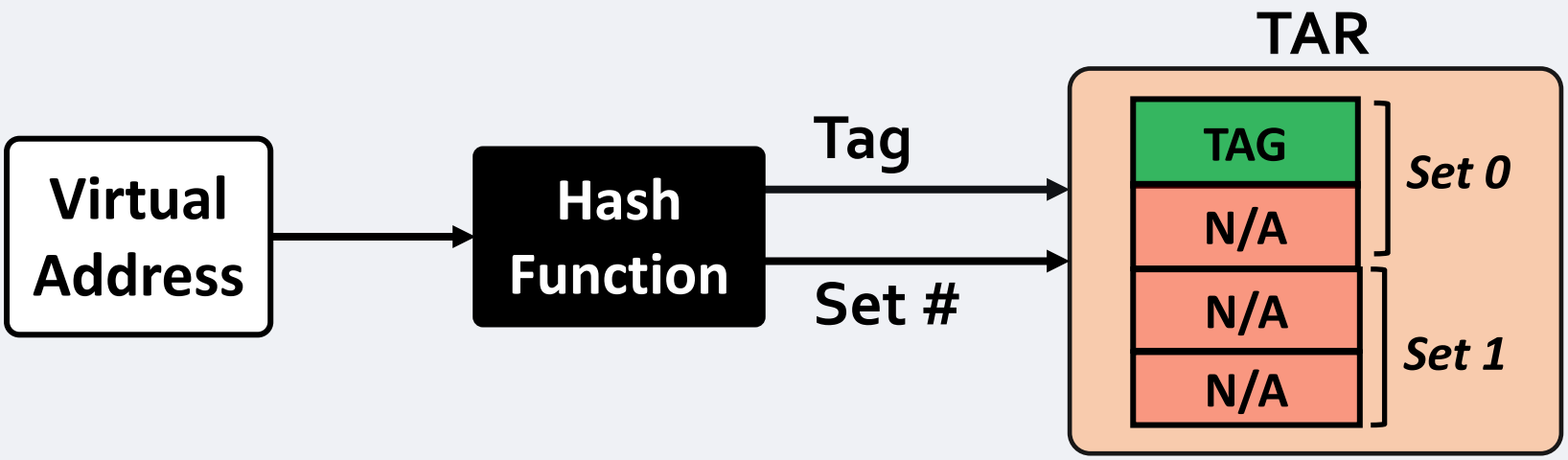
RSW Operations

1. *Tag Matching*

2. *Set Filtering*

RestSeg: Tag Matching

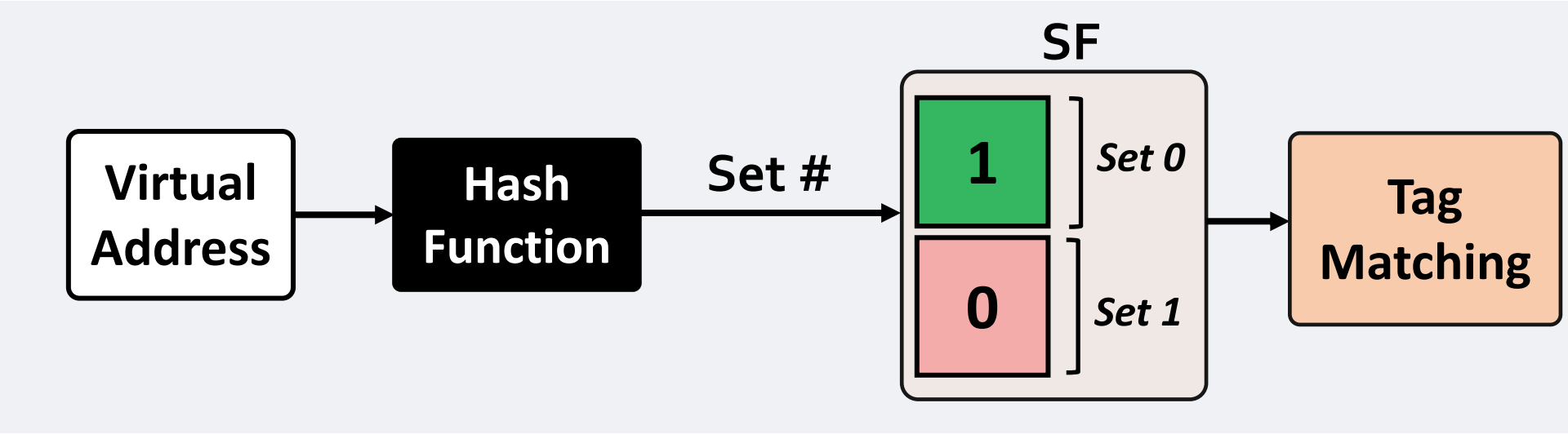
- **Tag matching** requires comparing the tags of all ways with the tag of the virtual page
- **Tag Array (TAR):** Array that stores the tag of each entry



Do we always have to do tag matching?

RestSeg: Set Filtering

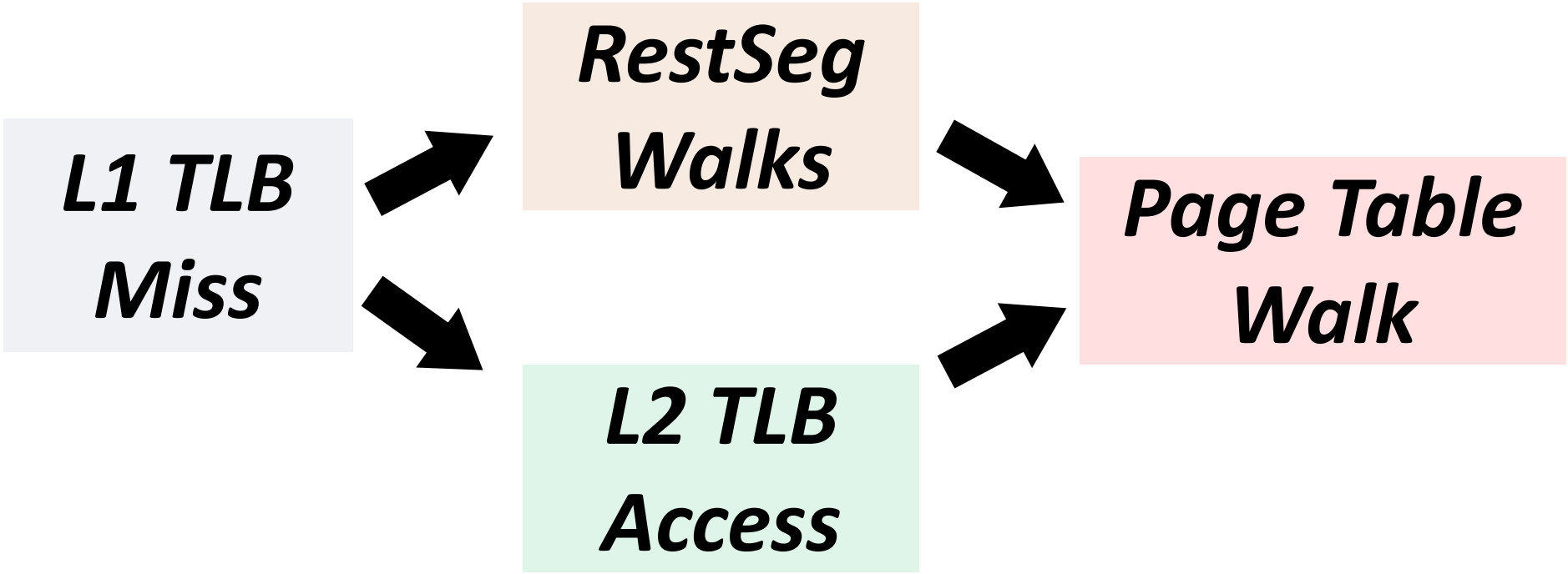
- **Set Filtering:** quickly discover if a set in the RestSeg is empty or not and filter tag mismatches
- **Set Filter (SF)**: Array of counters that keep track of the number of pages inside each set



Address Translation in Utopia

System employs:

- 2 RestSegs, one for 4KB and one for 2MB pages
- 1 FlexSeg



Talk Outline

Virtual Memory Background

Address Translation Overheads

Utopia: Hybrid Address Mappings

Utopia: Key Challenges

Evaluation Results

Utopia: Key Challenges

1. Which data should be placed in the RestSeg?

2. How to maintain RestSegs in the system

3. How to integrate Utopia in the address translation pipeline

Utopia: Key Challenges

1. **Which data should be placed in the RestSegs?**
2. How to maintain RestSegs in the system
3. How to integrate Utopia in the address translation pipeline

Page Placement in RestSeg

Our **goal** is to place costly-to-translate pages into a RestSeg

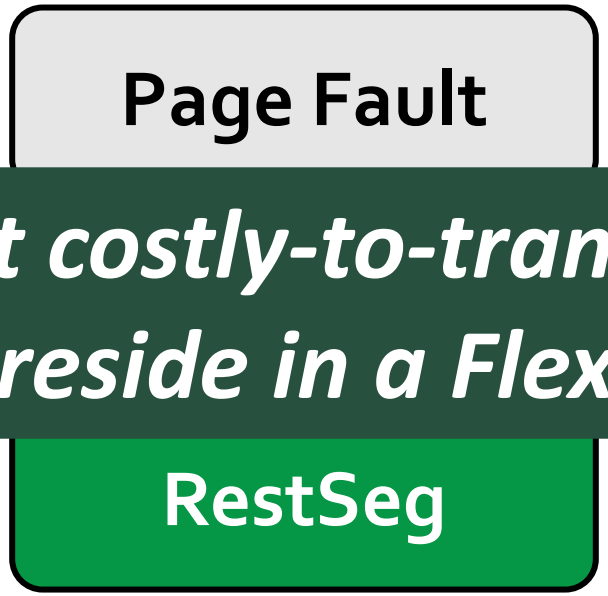
We propose two techniques to perform data placement in Utopia:

- **Page-Fault-based Allocation Policy**
- **PTW-Tracking-based Migration Policy**

Page-Fault-Based Allocation Policy

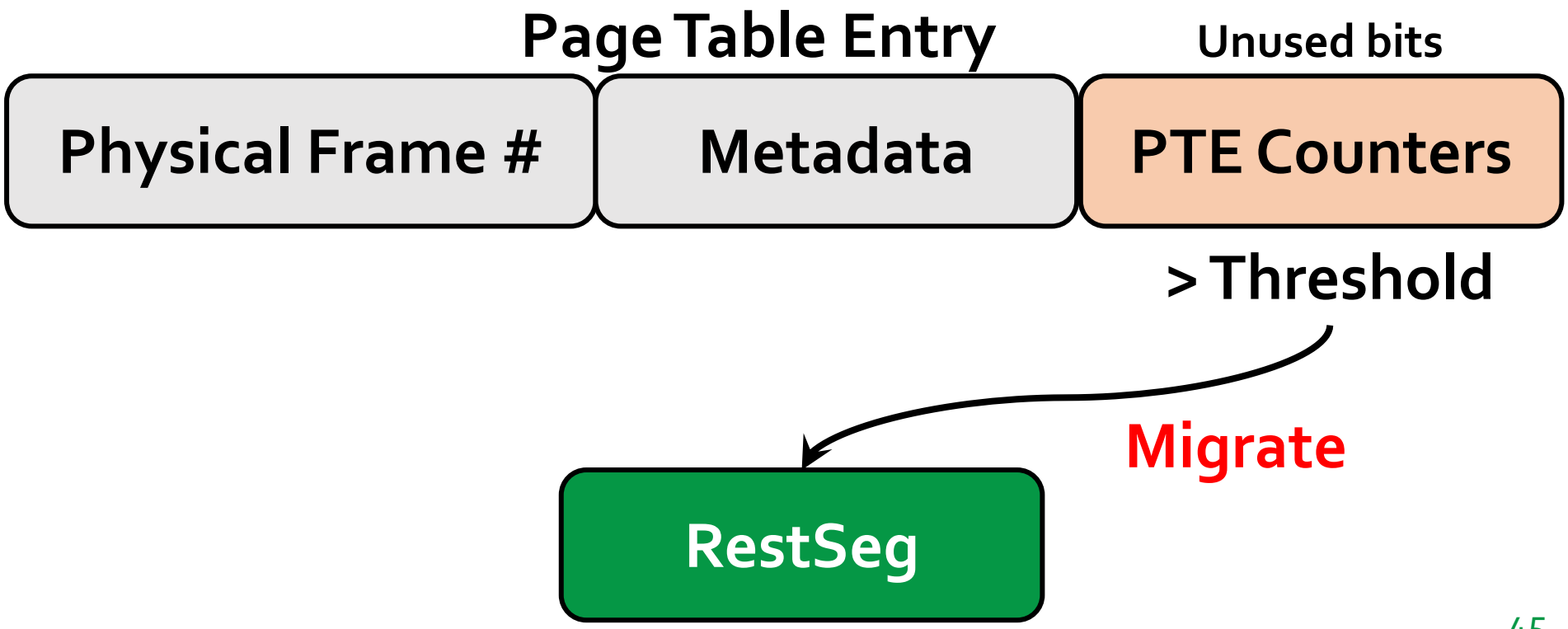
On a page fault

the page is directly allocated in a RestSeg



PTW-Tracking-Based Migration Policy

Use unused bits of each PTE as a counter that tracks the number and cost of PTWs for each page



Utopia: Three Key Challenges

1. Which data should be placed in the RestSegs?

2. **How to maintain RestSegs in the system**

3. How to integrate Utopia in the address translation pipeline

OS Support for Utopia

OS supports Utopia in three ways by handling:

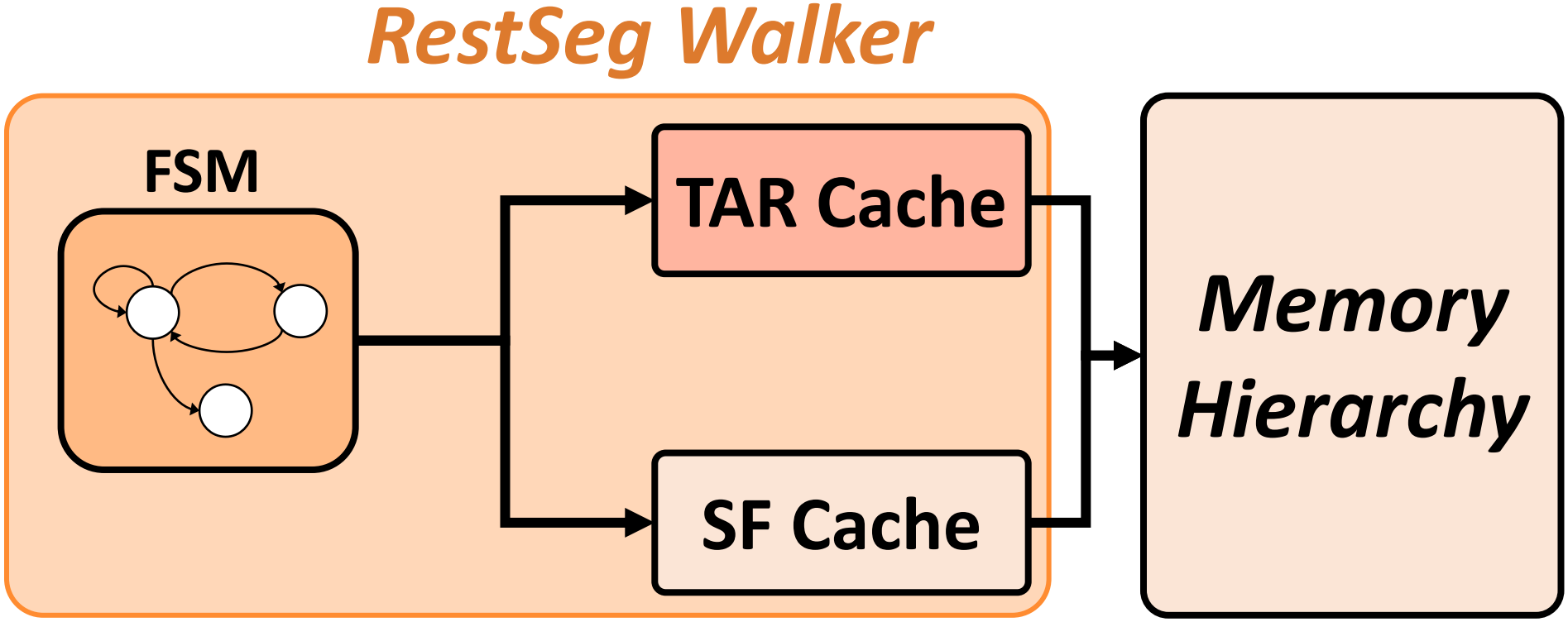
1. **Allocations** in a RestSeg
2. **Replacements** in a RestSeg
3. **Migrations** to/from a RestSeg

Detailed description in the paper

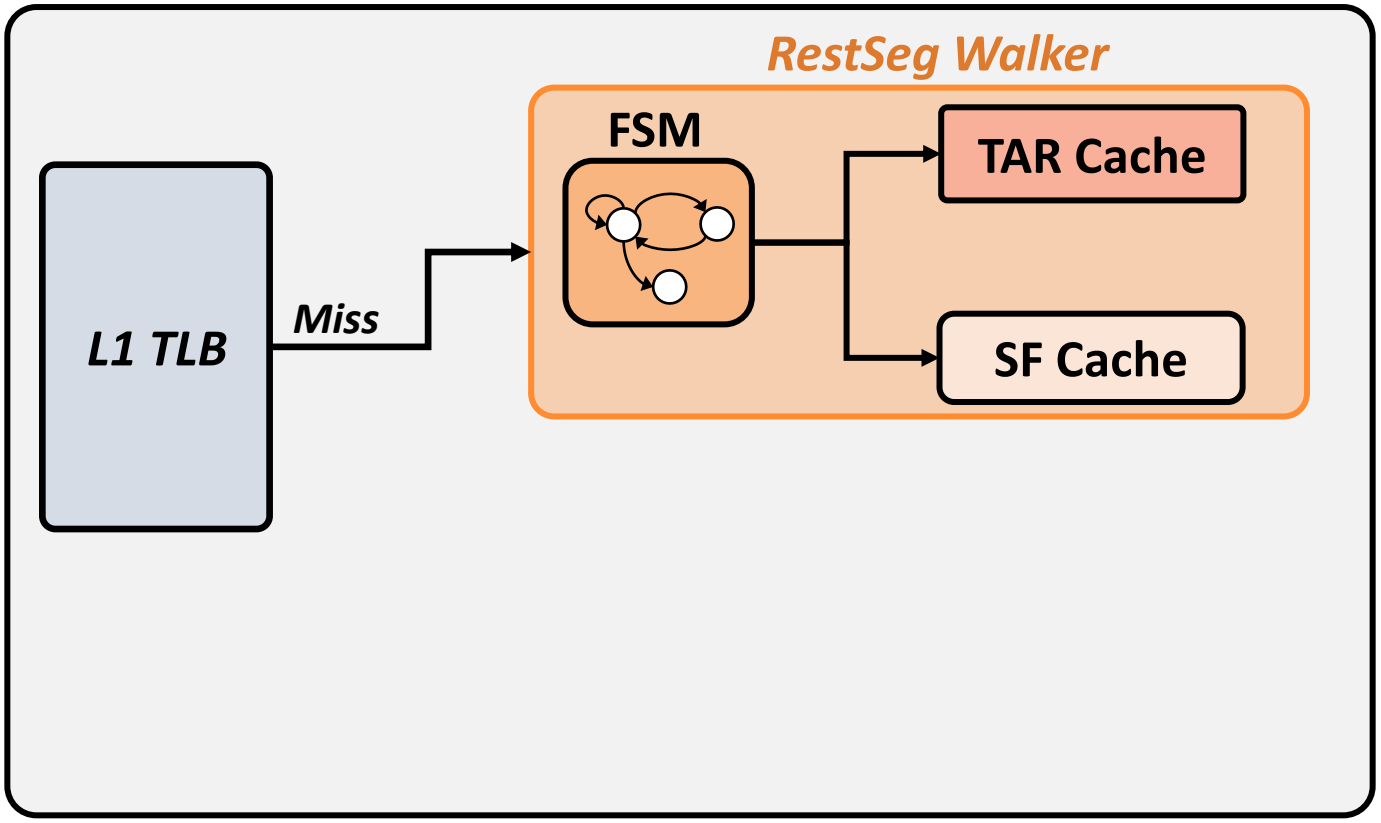
Utopia: 3 Key Challenges

- Which data should be placed in the RestSegs?
- How to maintain RestSegs in the system
- **How to integrate Utopia in the address translation pipeline**

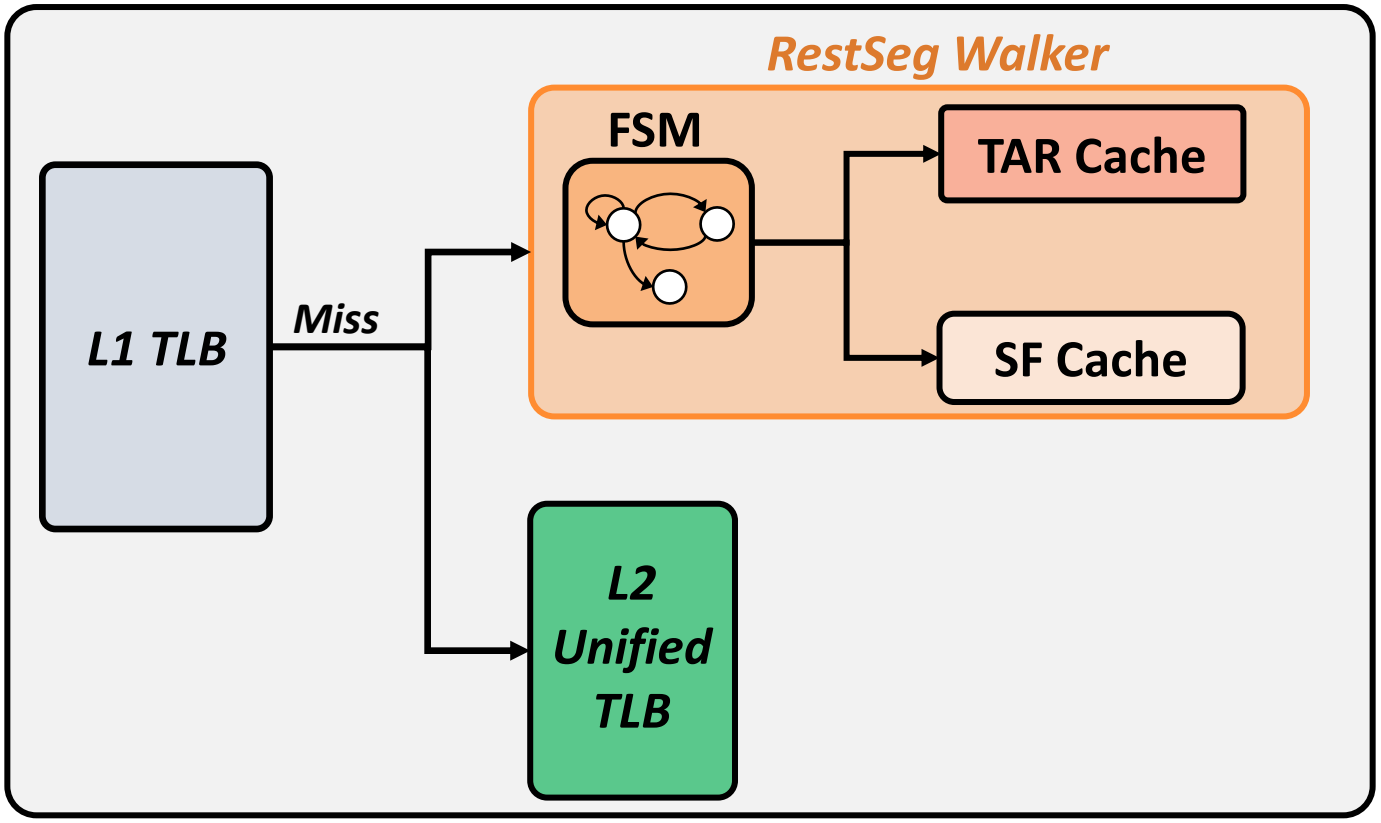
RestSeg Walker in MMU



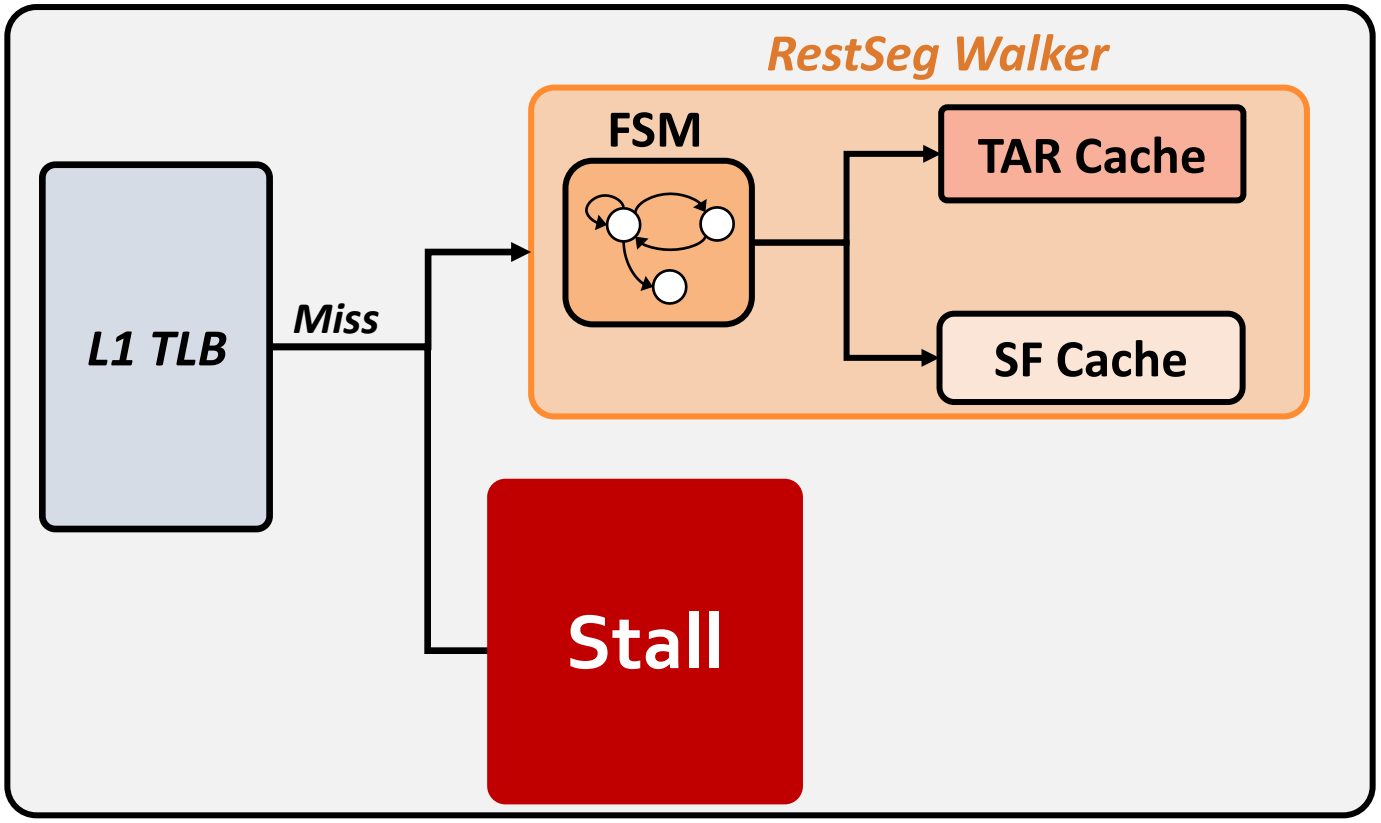
MMU: Page inside RestSeg (I)



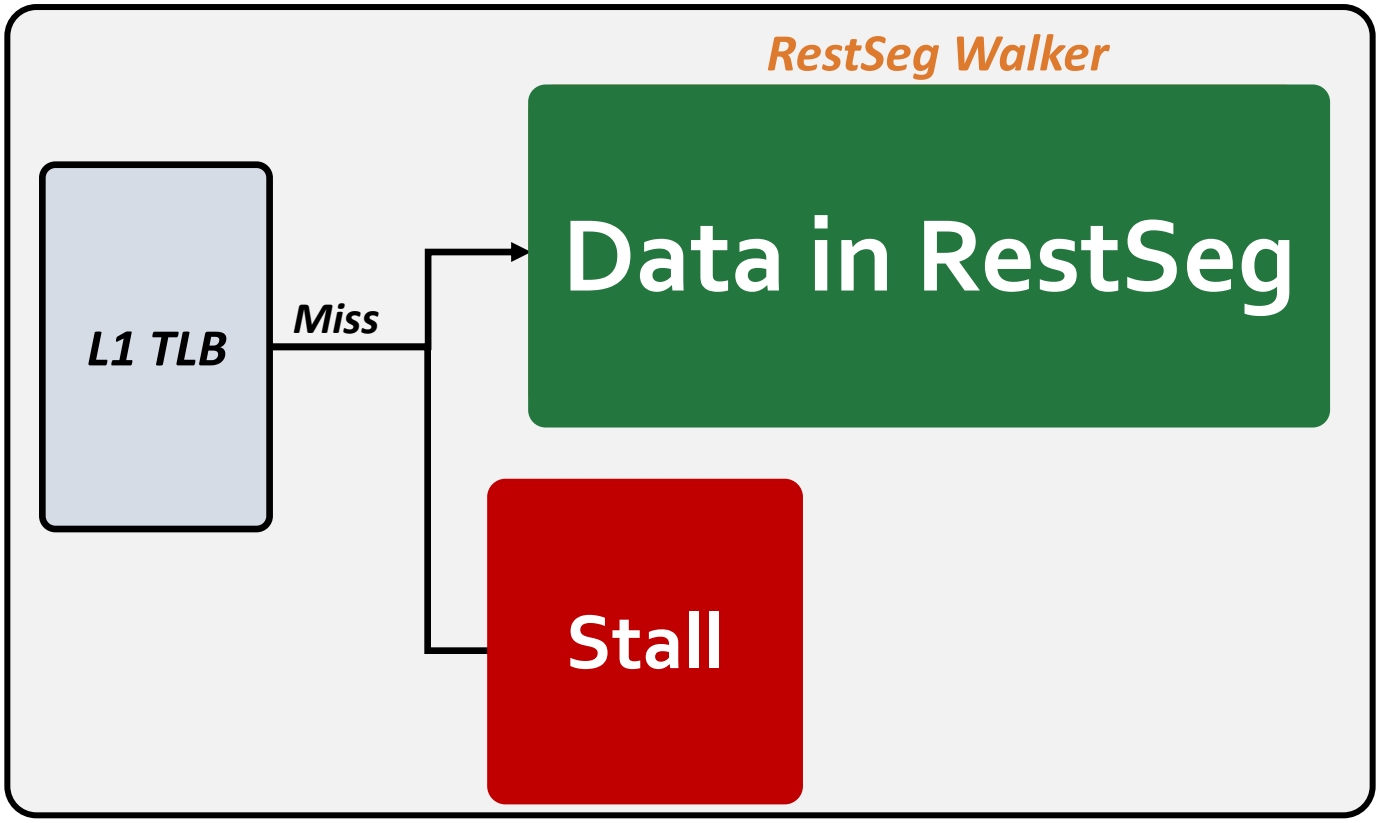
MMU: Page inside RestSeg (I)



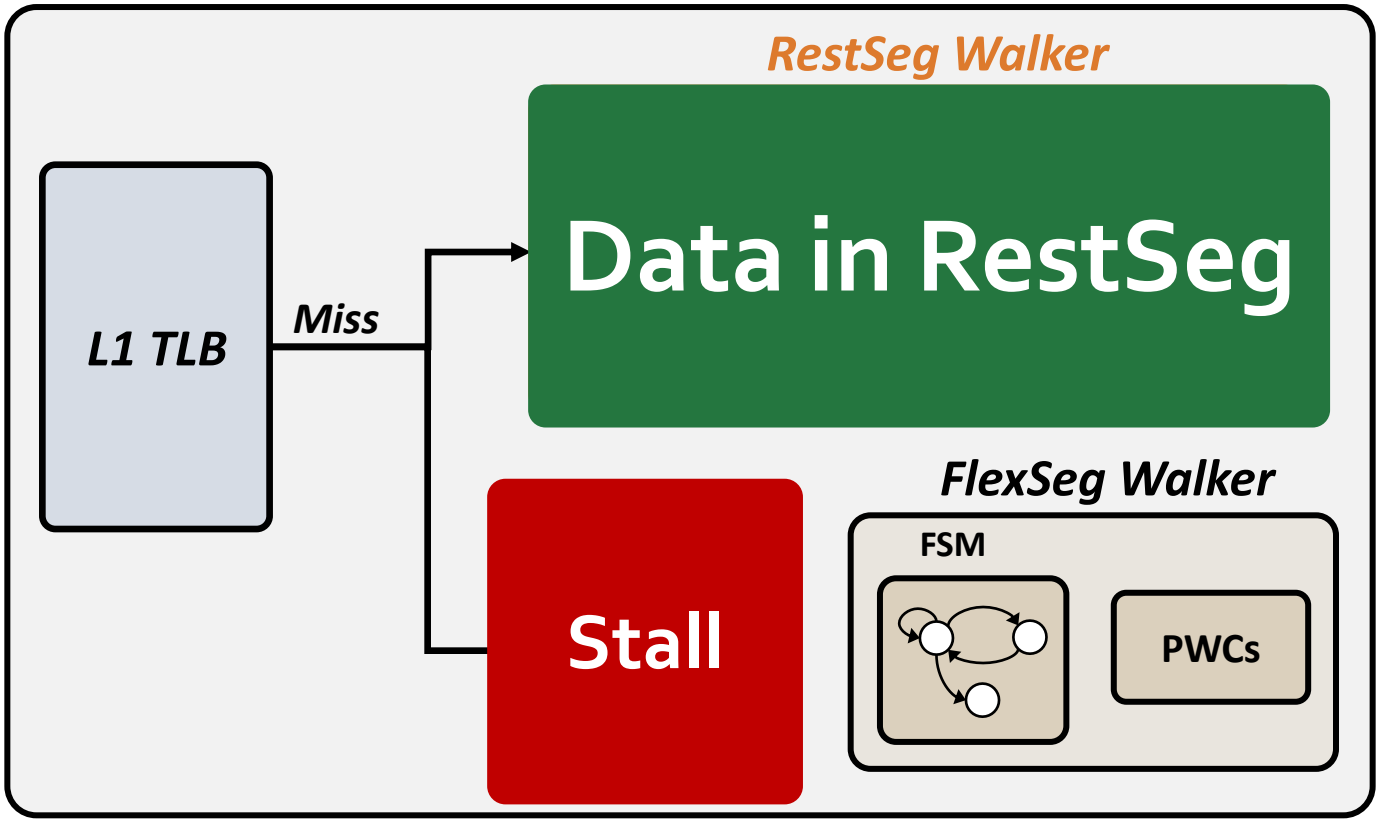
MMU: Page inside RestSeg (I)



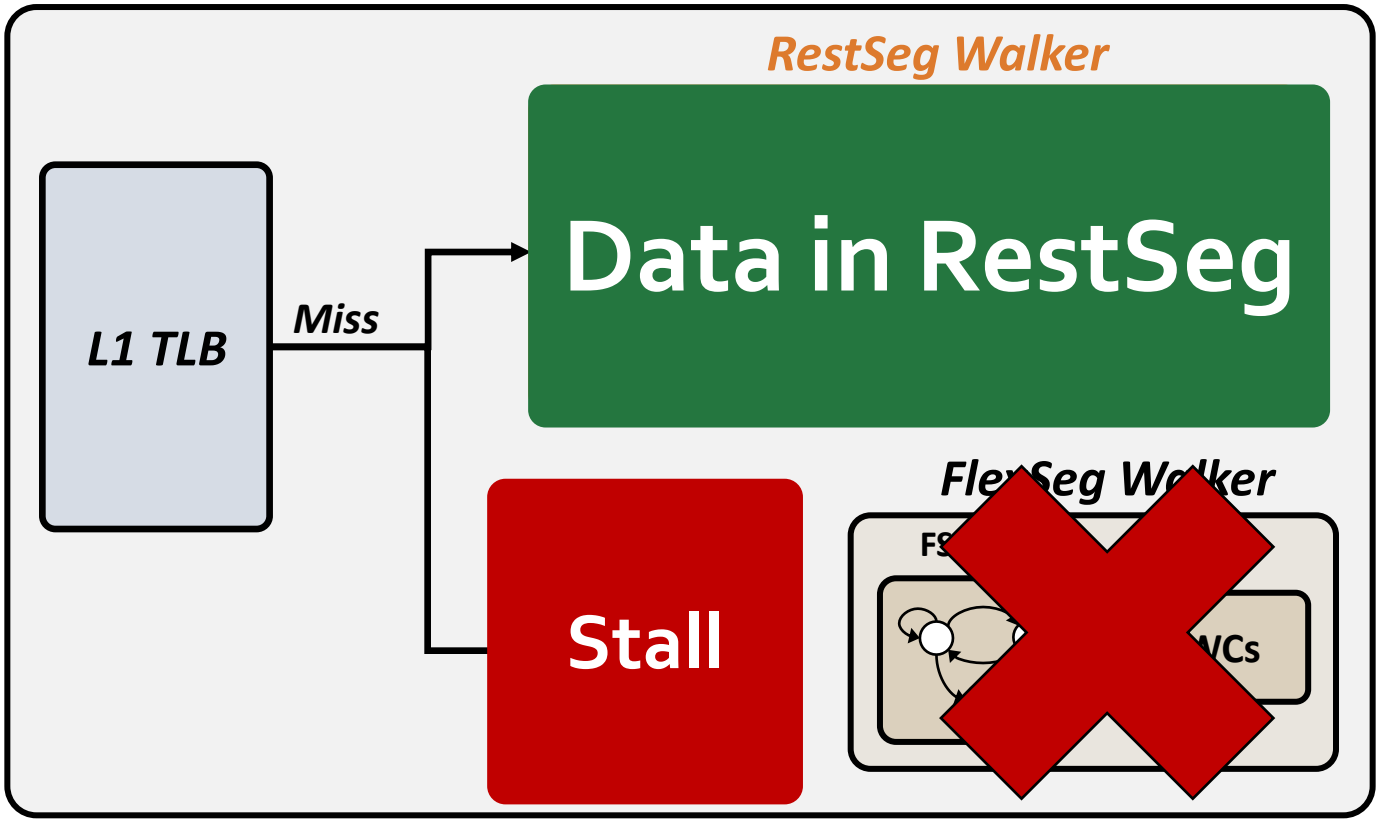
MMU: Page inside RestSeg (I)



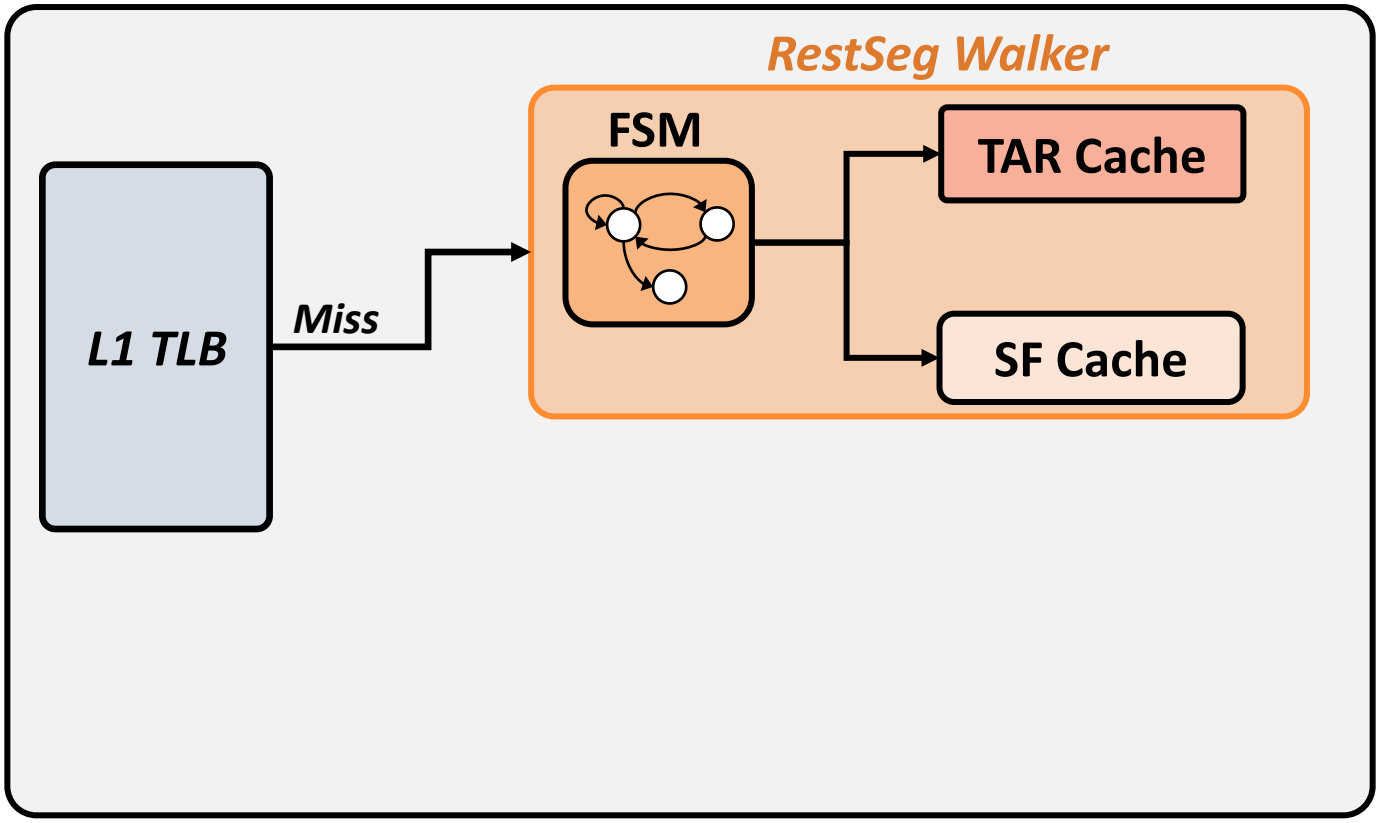
MMU: Page inside RestSeg (I)



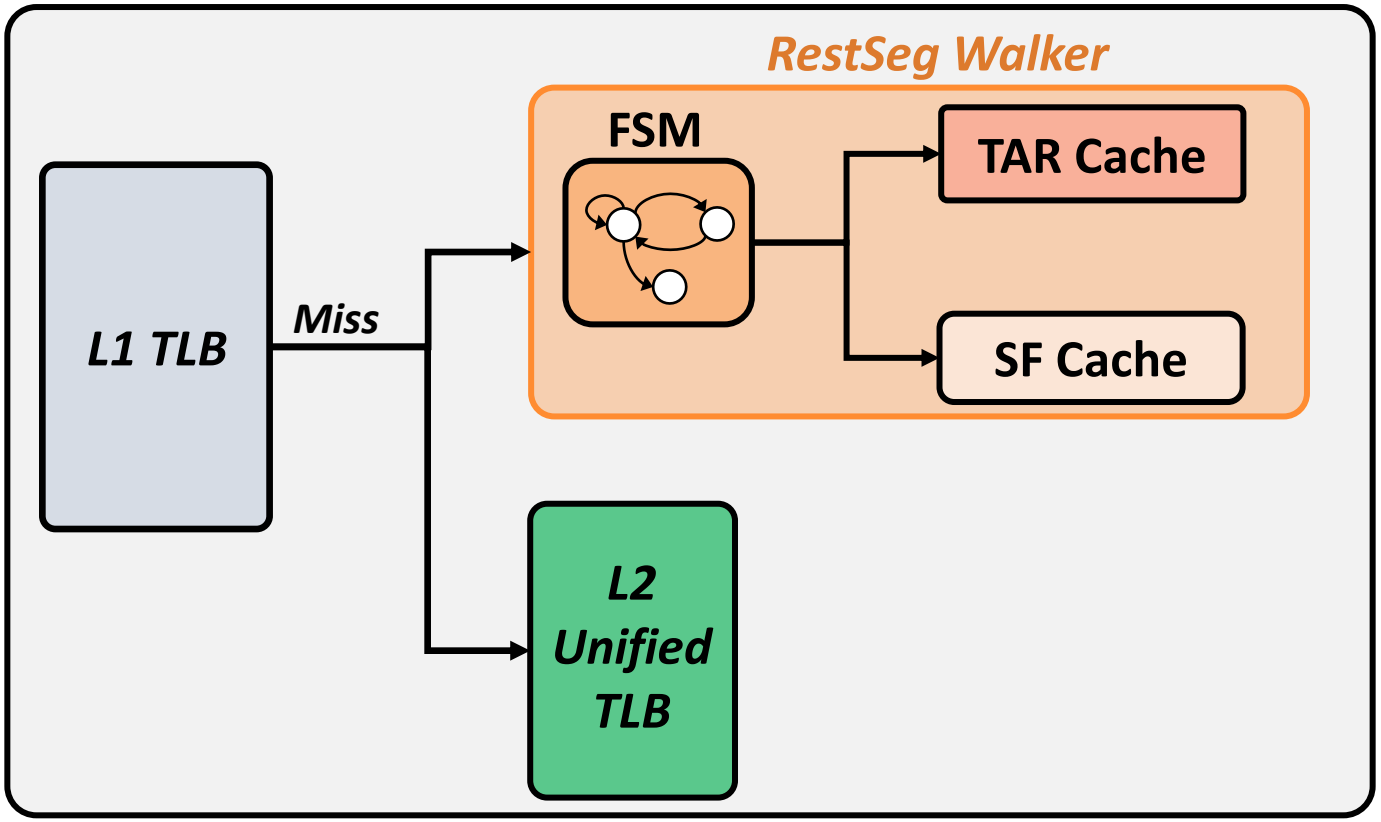
MMU: Page inside RestSeg (I)



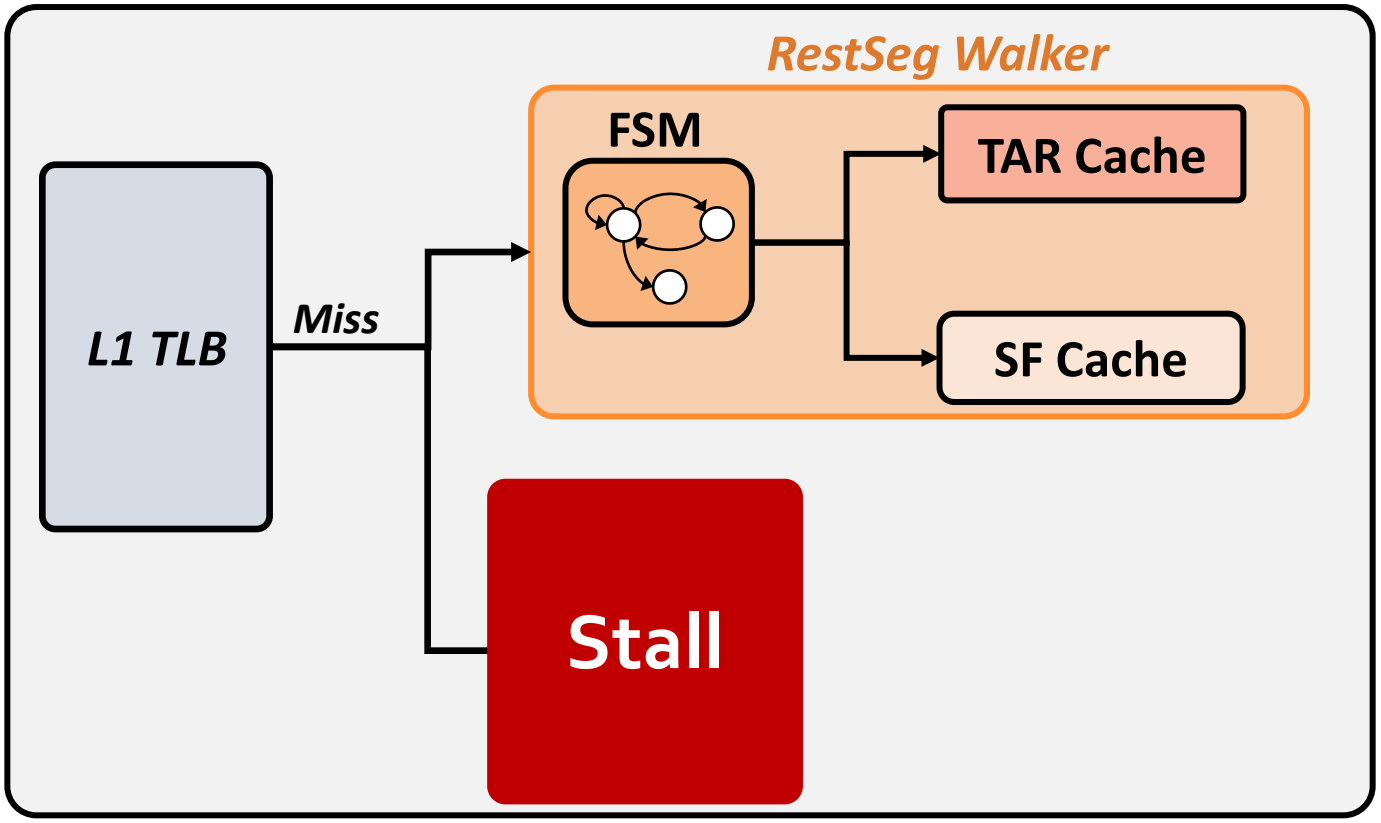
MMU: Page inside FlexSeg (II)



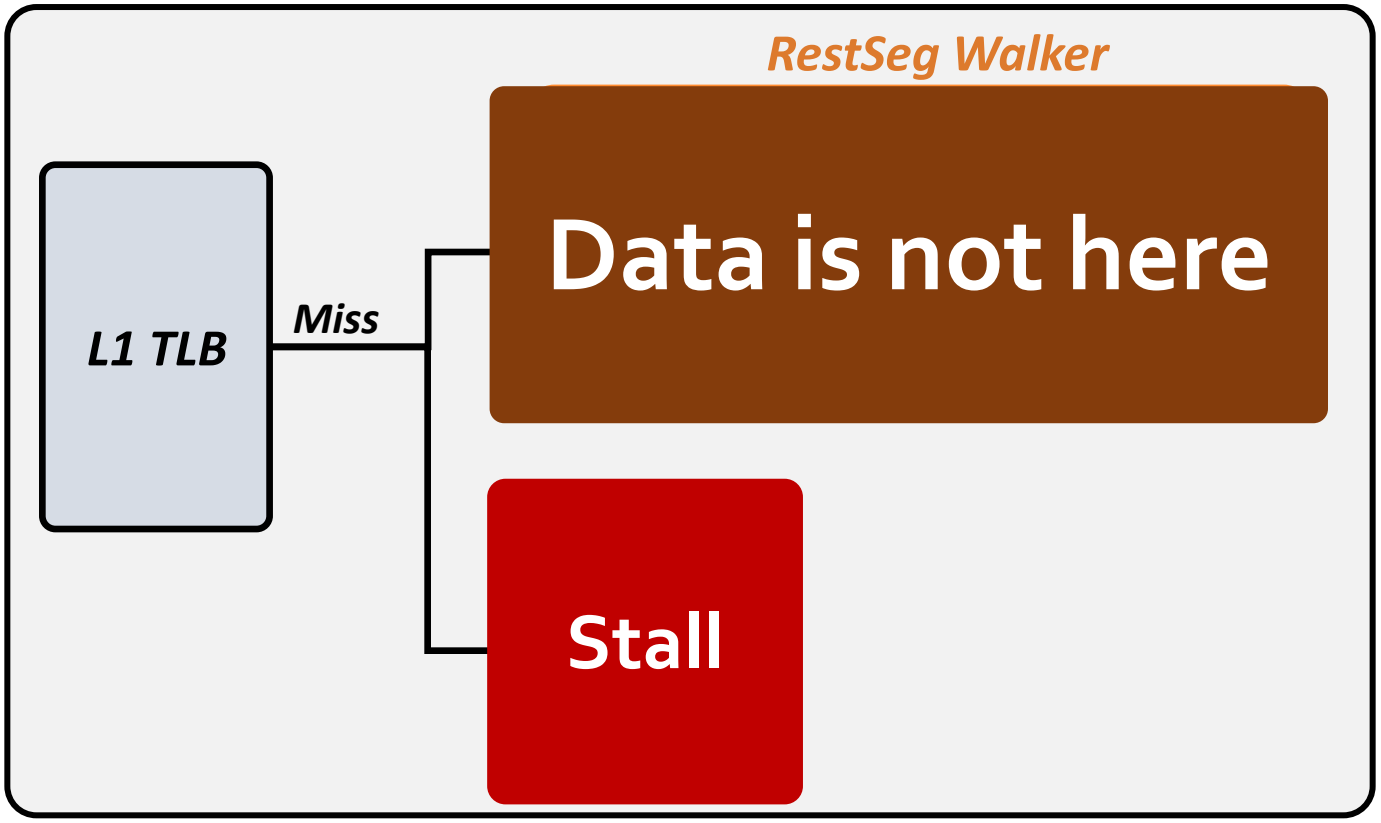
MMU: Page inside FlexSeg (II)



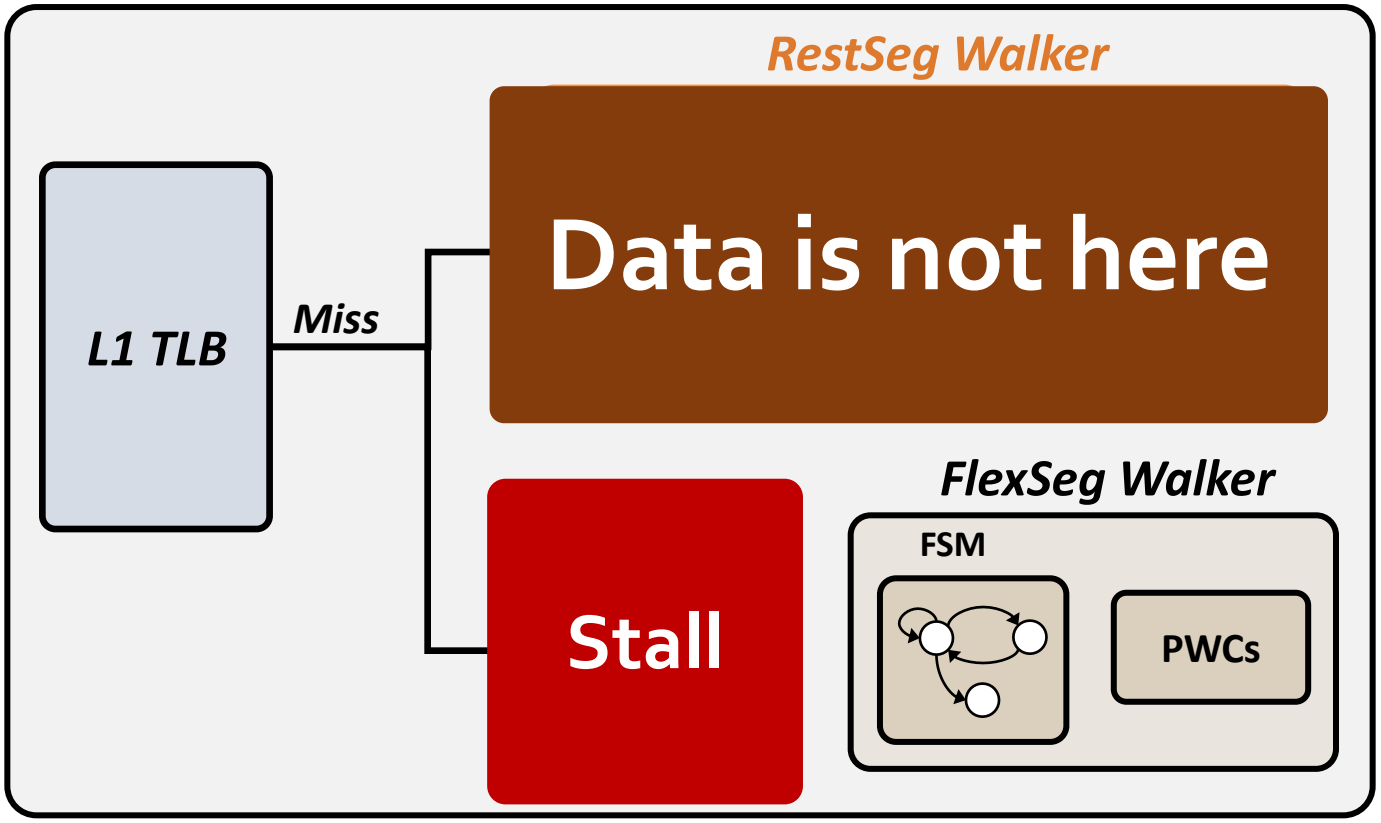
MMU: Page inside FlexSeg (II)



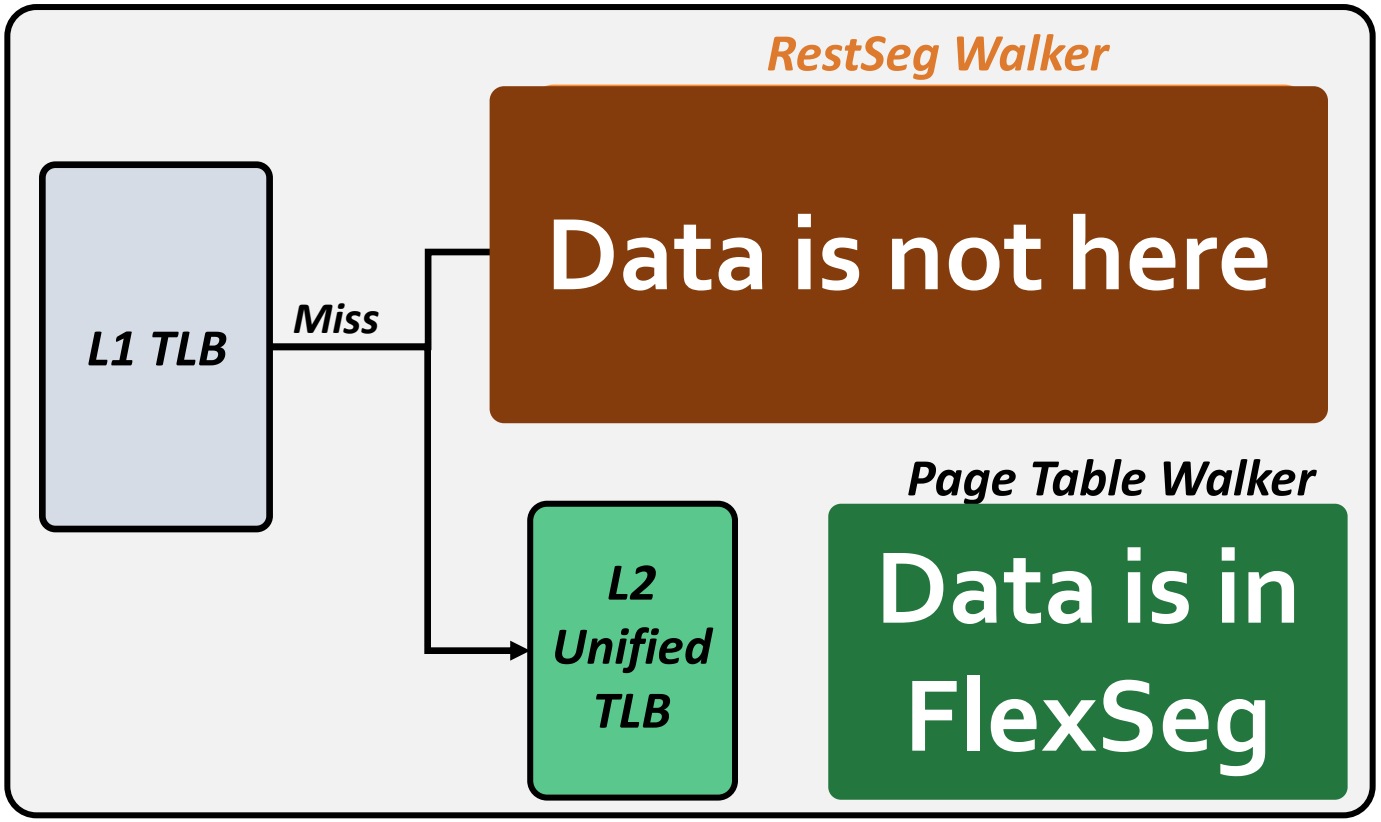
MMU: Page inside FlexSeg (II)



MMU: Page inside FlexSeg (II)



MMU: Page inside FlexSeg (II)



Area & Power Overhead

- Area and power overhead evaluation using McPat
- Comparison to a high-end Intel Raptor Lake

Utopia incurs 0.64% area and 0.72% power overhead per core

Talk Outline

Virtual Memory Background

Address Translation Overheads

Utopia: Hybrid Address Mappings

Utopia: Key Challenges

Evaluation Results

Evaluation Methodology

- Sniper Multicore Simulator extended with:
  - Page table walker & page walk caches
  - Buddy allocator
  - Migration Latency

Our poster at MICRO 2023 SRC introduces a new open-source simulation framework for VM research

https://github.com/CMU-SAFARI/Virtuoso

Evaluation Methodology

- Sniper Multicore Simulator extended with:
  - Page table walker & page walk caches
  - Buddy allocator
  - Migration Latency

- Workloads**: Executed for 500M instructions**

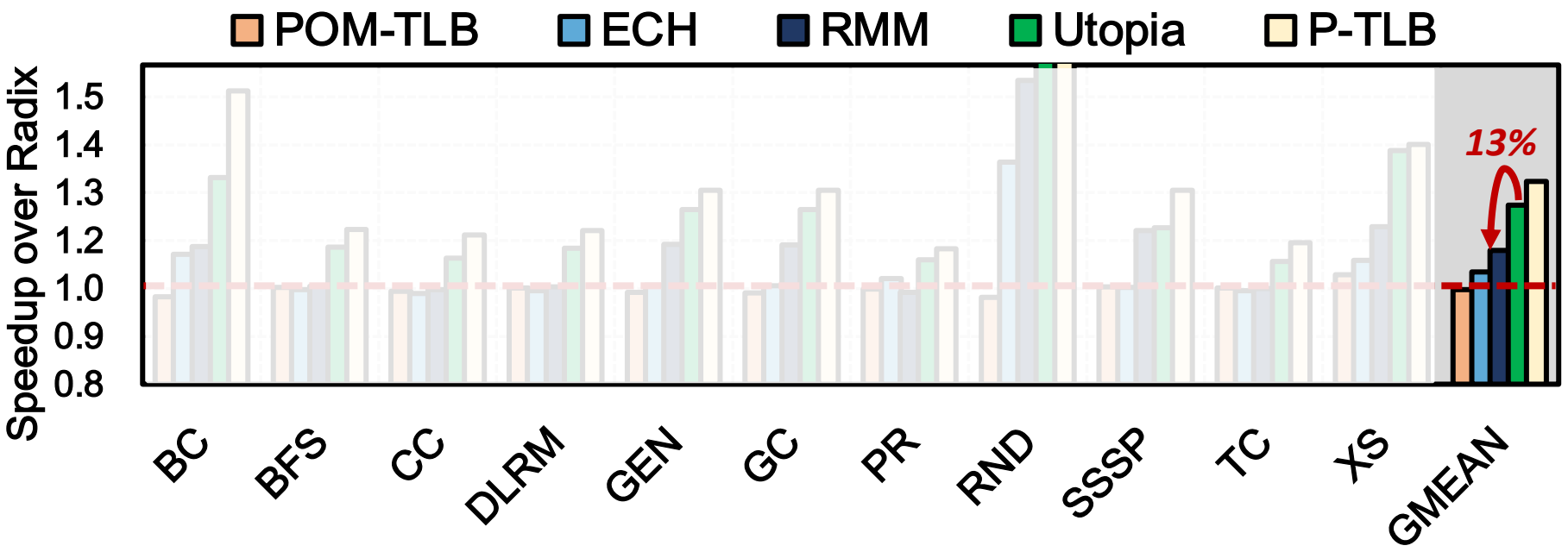
  - GraphBIG: PR, BFS, BC, GC, CC
  - HPCC: Randacc
  - XSBench: Particle Simulation with 15K grid
  - DLRM: SLS-like
  - GenomicsBench: k-mer count

Evaluated Mechanisms

- Baseline with **Radix PT** and **Transparent Huge Pages** enabled
- **POM-TLB**[1]: State-of-the art large **software-managed TLB**
- **ECH**[2]: State-of-the-art **hash-based page table**
- **RMM**[3]**: Contiguity-aware** address translation
- **Utopia:** 512MB RestSegs (one for 4KB and one for 2MB pages)
- **P-TLB**: Perfect L1 TLB (translation requests always hit in L1 TLB)

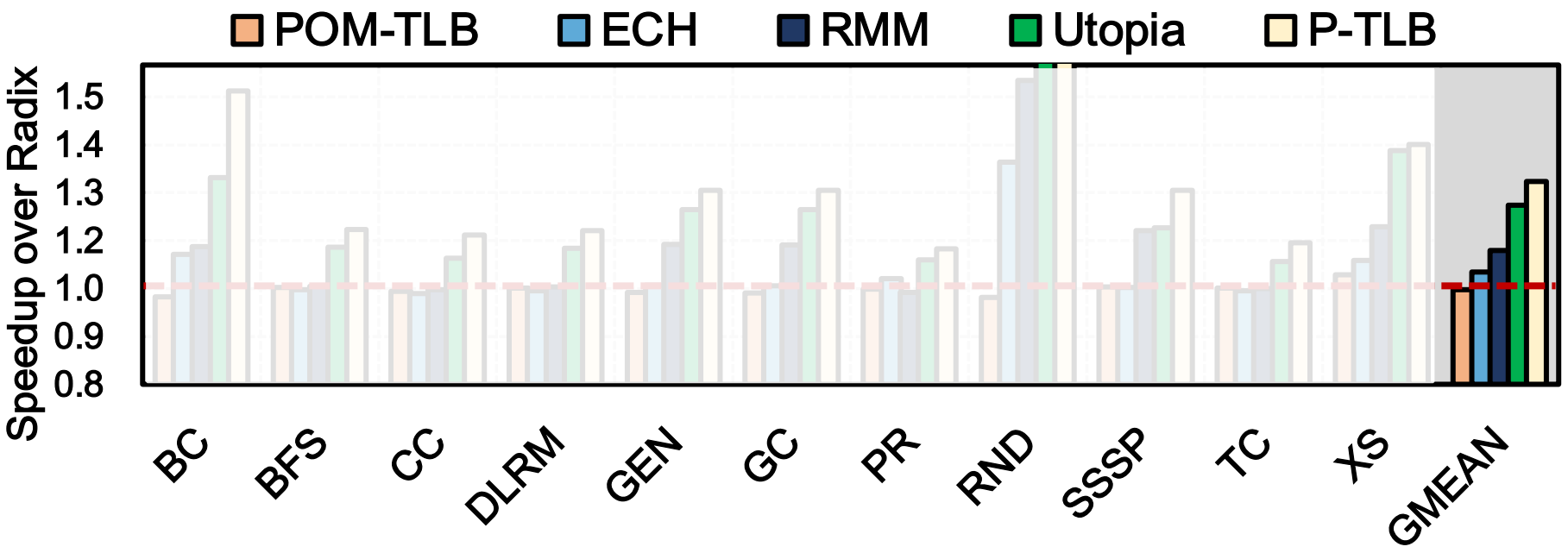
[1] Ryoo et al. "Rethinking TLB designs in virtualized environments: A very large part-of-memory TLB" ISCA '17
[2] Skarlatos et al. "Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism" ASPLOS '20
[3] Karakostas et al. "Redundant memory mappings for fast access to large memories" ISCA '15

Performance Results



Utopia outperforms the second-best performing scheme (RMM) by 13% and Radix by 24%

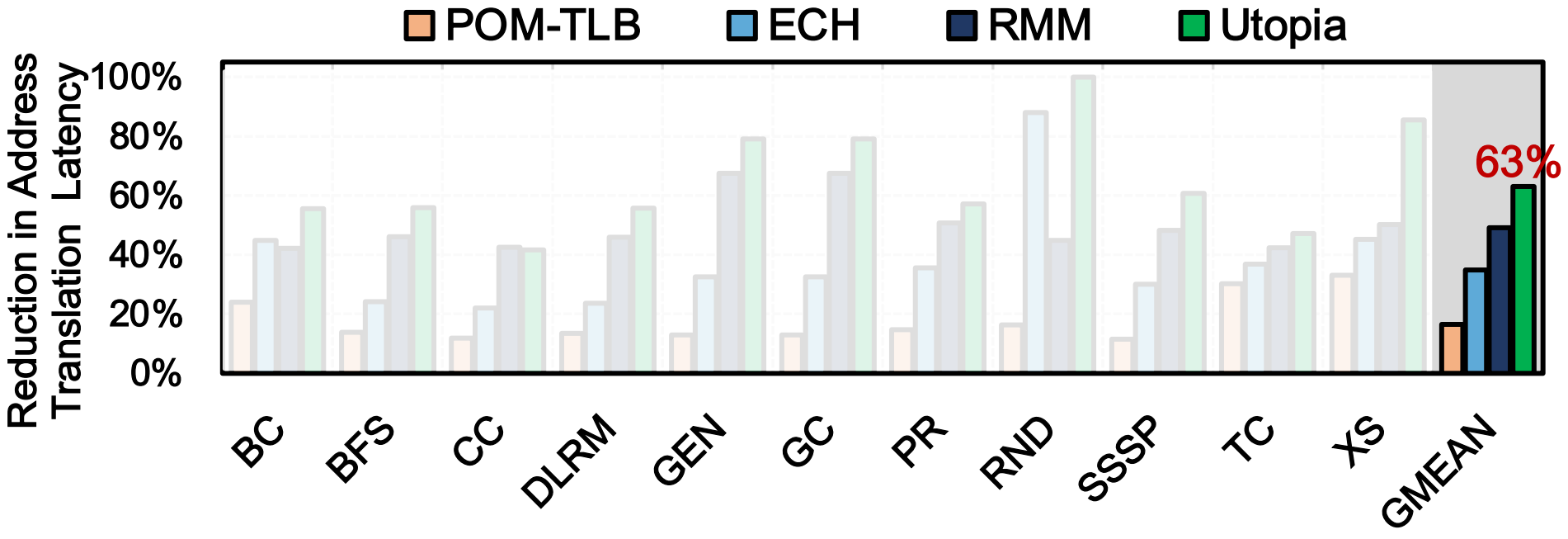
Performance Results



Utopia outperforms the second-best performing scheme (RMM) by 13% and Radix by 24%

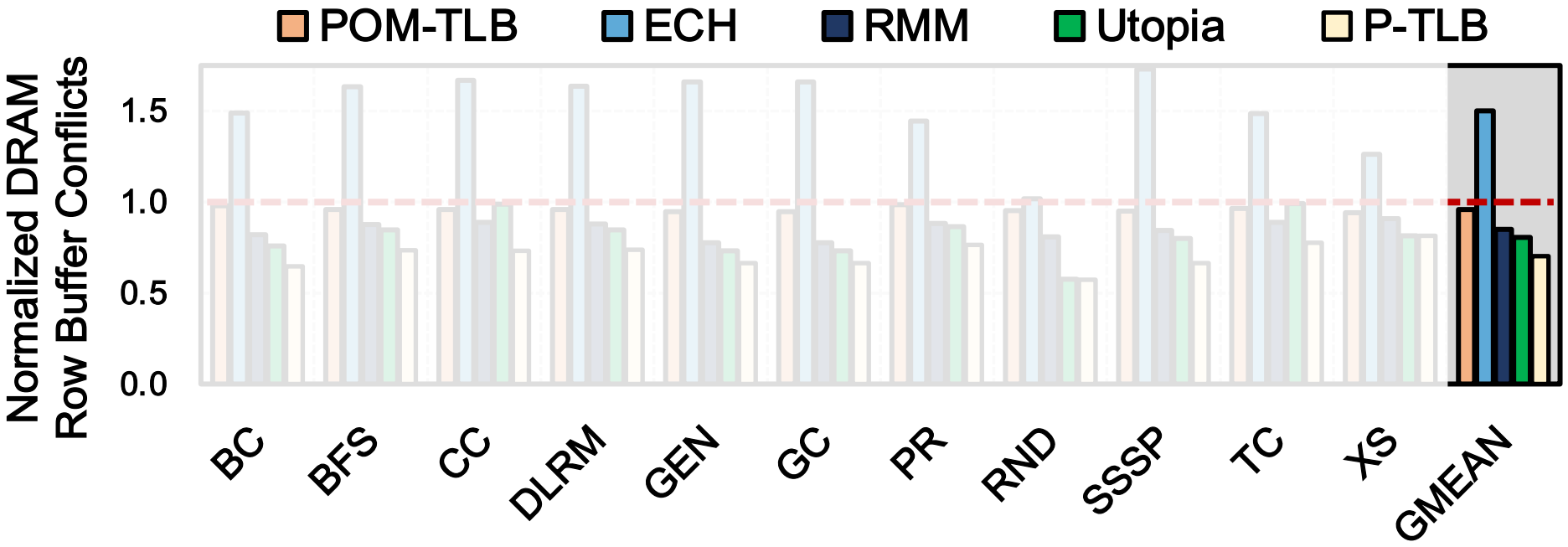
Utopia's performance is within 95% of P-TLB

Translation Latency



Utopia reduces average translation latency by 63% over Radix and 14% over RMM

Main Memory Interference



Utopia reduces row buffer conflicts by 20% over Radix and 9% less than P-TLB

More Results and Details in the Paper

- Sensitivity across different hash functions
- Sensitivity to parallel/serial TLB access and RSW
- Sensitivity to RestSeg size
- Reuse-level distribution of 4KB pages
- Effect of migration to memory requests
- TAR & SF cache hit rate
- Overhead across different context-switch quanta

https://arxiv.org/abs/2211.12205

More Results and Details in the Paper

- Se
- Se
- Se
- Re
- Ef
- TA
- Ov

RSW

uanta

Utopia: Fast and Efficient Address Translation via Hybrid Restrictive & Flexible Virtual-to-Physical Address Mappings

Konstantinos Kanellopoulos[1] Rahul Bera[1] Kosta Stojiljkovic[1] Nisa Bostanci[1] Can Firtina[1]
Rachata Ausavarungnirun[2] Rakesh Kumar[3] Nastaran Hajinazar[4] Mohammad Sadrosadati[1]
Nandita Vijaykumar[5] Onur Mutlu[1]

[1]ETH Zürich [2]King Mongkut's University of Technology North Bangkok
[3]Norwegian University of Science and Technology [4]Intel Labs [5]University of Toronto

Abstract

Conventional virtual memory (VM) frameworks enable a virtual address to flexibly map to *any* physical address. This flexibility necessitates large data structures to store virtual-to-physical mappings, which leads to high address translation latency and large translation-induced interference in the memory hierarchy, especially in data-intensive workloads. On the other hand, restricting the address mapping so that a virtual address can only map to a specific set of physical addresses can significantly reduce address translation overheads by making use of compact and efficient translation structures. However, restricting the address mapping flexibility across the entire main memory severely limits data sharing across different processes and increases data accesses to the swap space of the storage device even in the presence of free memory.

We propose *Utopia*, a new hybrid virtual-to-physical address mapping scheme that allows *both* flexible and restrictive hash-based address mapping schemes to harmoniously *co-exist* in the system. The key idea of Utopia is to manage physical memory using two types of physical memory segments: restrictive segments and flexi-

1 Introduction

Virtual memory (VM) serves as a foundational element in most computing systems, simplifying the programming model by offering an abstraction layer over physical memory [2–24]. In the presence of VM, the operating system (OS) maps each virtual address to its corresponding physical memory address to facilitate application-transparent memory management, process isolation, and memory protection. The virtual-to-physical mapping scheme in conventional VM frameworks allows a virtual address to flexibly map to *any* physical address. This flexibility enables key VM functionalities, such as (i) data sharing between processes while maintaining process isolation and (ii) avoiding frequent swapping (i.e., avoiding storing data in the swap space of the storage device in the presence of free main memory space). However, a flexible mapping scheme requires mapping metadata for every virtual address and its corresponding physical address, which is stored in the page table (PT). As shown in multiple prior works [25–35], data-intensive workloads do not efficiently use translation-dedicated hardware structures and the processor performs frequent PT accesses, i.e., a process called

https://arxiv.org/abs/2211.12205

Conclusion

We propose **Utopia**, a new virtual-to-physical mapping scheme that enables both:

Restrictive Mapping **Flexible Mapping**

Harmoniously co-exist in the system

Utopia achieves (i) **13% higher performance** than the state-of-the-art **contiguity-aware translation scheme** and (ii) **95% of the performance** of an ideal **perfect-TLB**

Utopia is open source

https://github.com/CMU-SAFARI/Utopia

Arxiv

UTOPIA

Fast and Efficient Address Translation via Hybrid Restrictive & Flexible Virtual-to-Physical Address Mappings

https://github.com/CMU-SAFARI/Utopia

Github

Konstantinos Kanellopoulos

Rahul Bera, Kosta Stojiljkovic, Nisa Bostanci, Can Firtina, Rachata Ausavarungnirun, Rakesh Kumar, Nastaran Hajinazar, Mohammad Sadrosadati, Nandita Vijaykumar, and Onur Mutlu

SAFARI Research Group
safari.ethz.ch

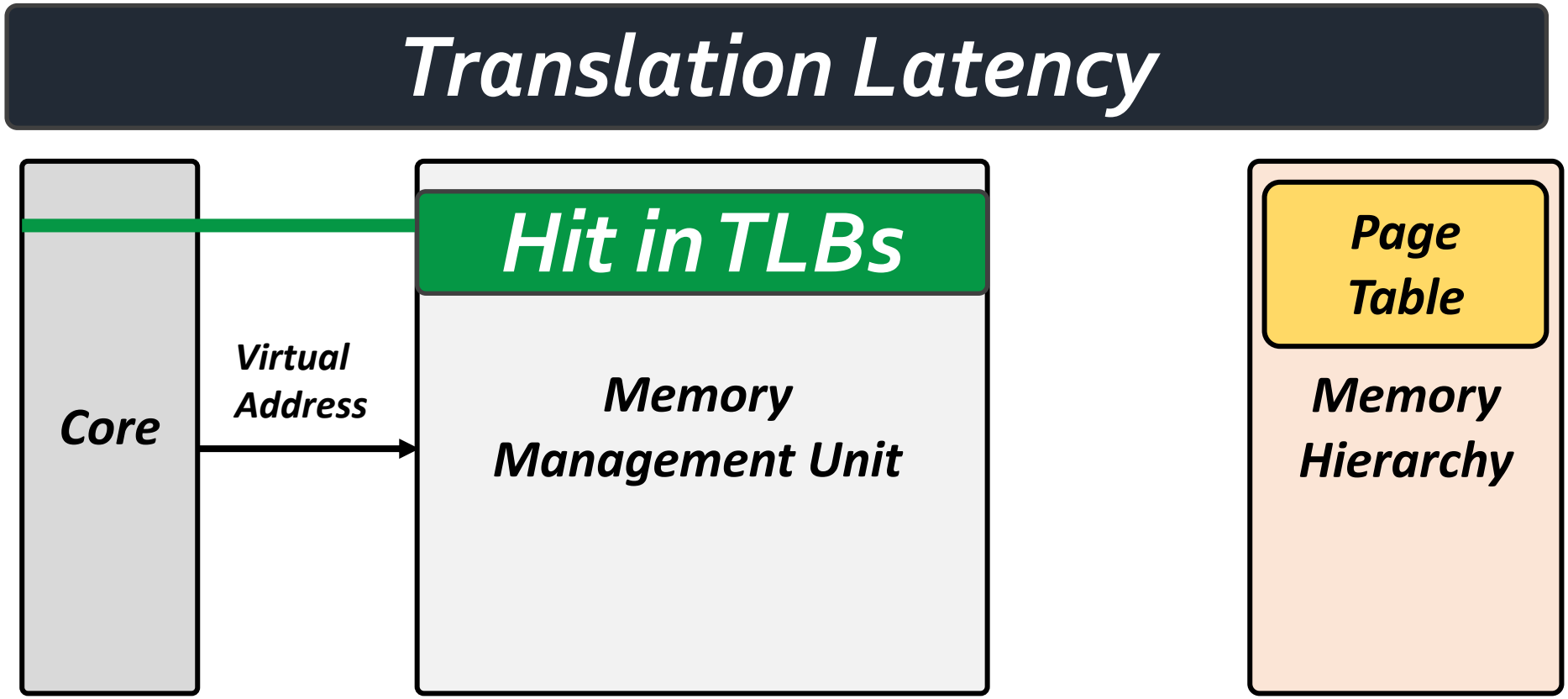
ETH zürich

intel

UNIVERSITY OF TORONTO

NTNU

Address Translation Flow (III)



Virtual Memory Basics

Virtual Memory (VM) is one of the **cornerstones** of most modern computing systems

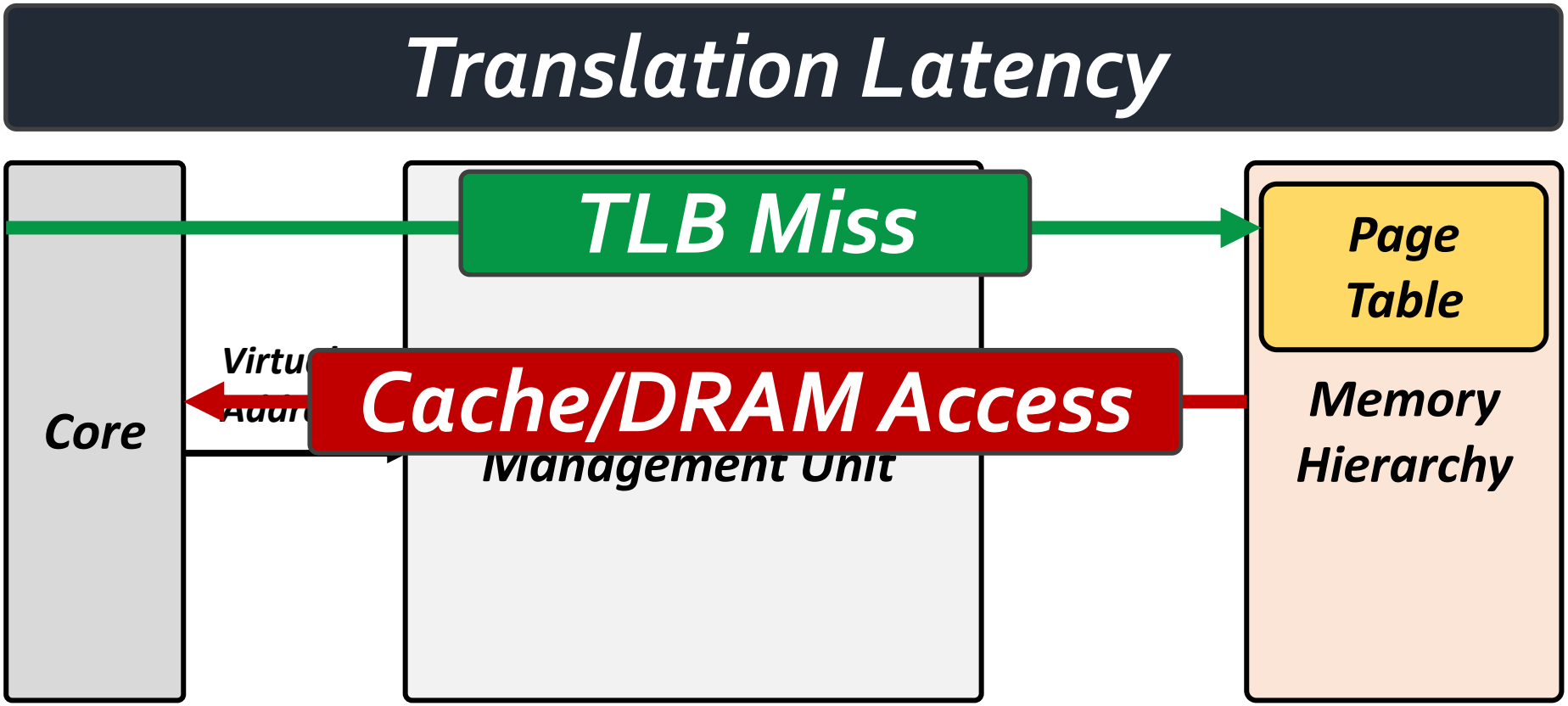
Conventional VM designs provide :

(1) Application-transparent **memory management**

(2) Data **sharing**

(3) Process **isolation**

Address Translation Flow (III)



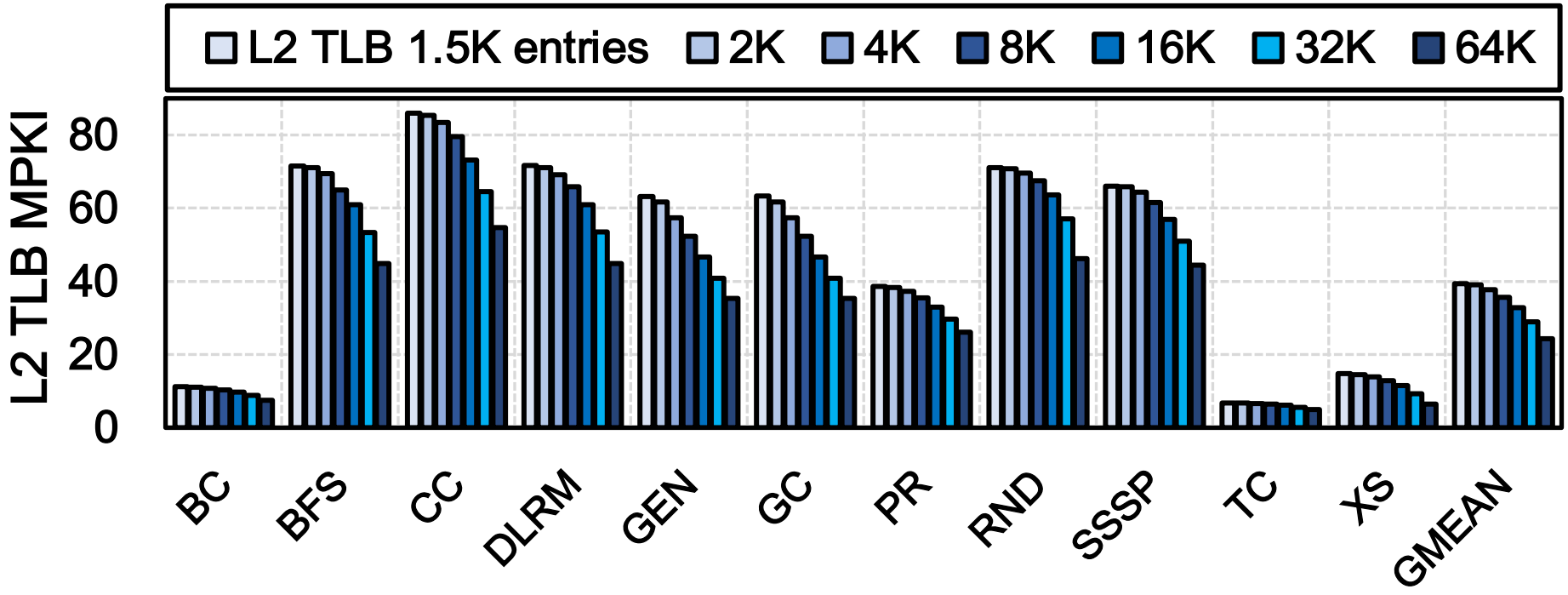
High Latency PTWs

Irregular Memory Access Patterns

Large Datasets

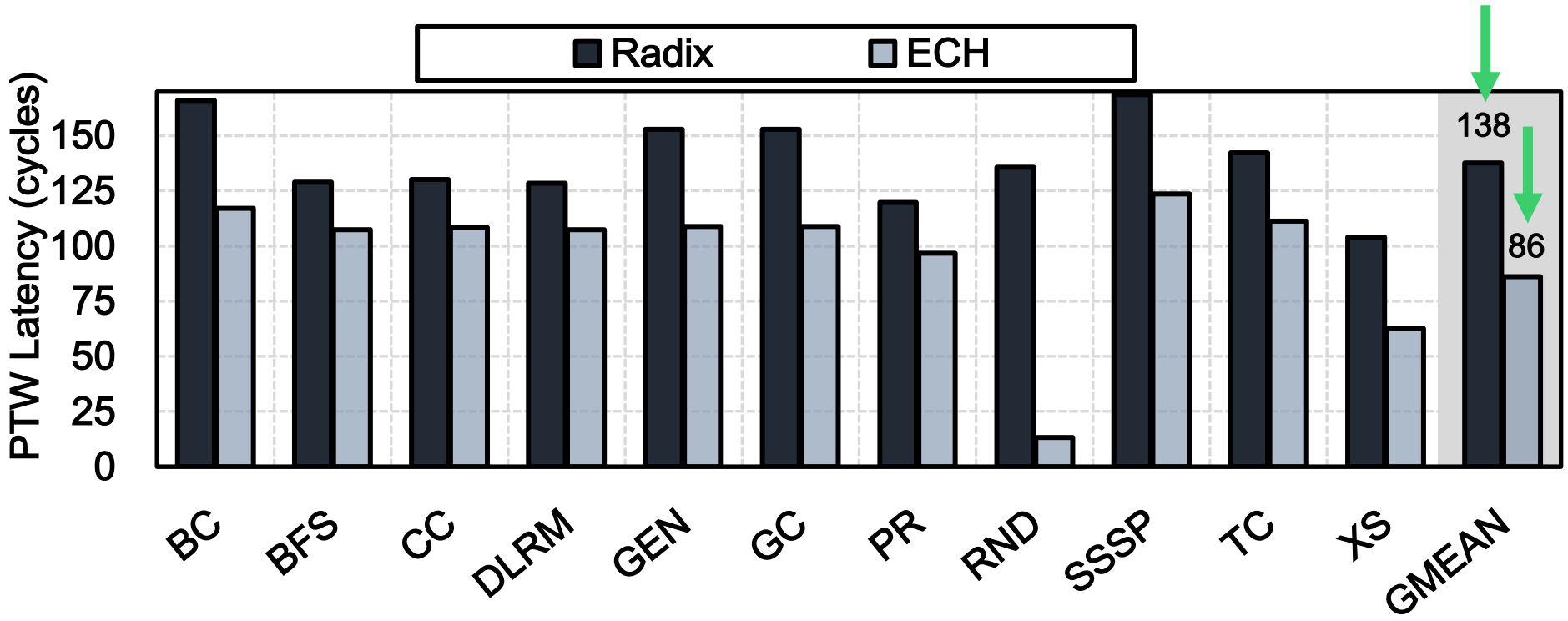
Large and costly-to-access PT

Frequent PTWs



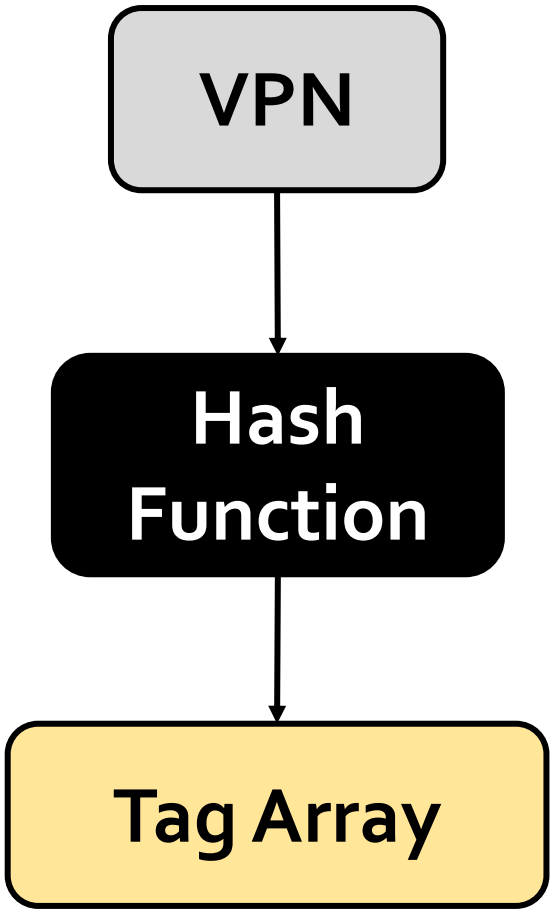
Even the largest 64K-entry L2 TLB experiences 24 MPKI on average

High Latency PTWs



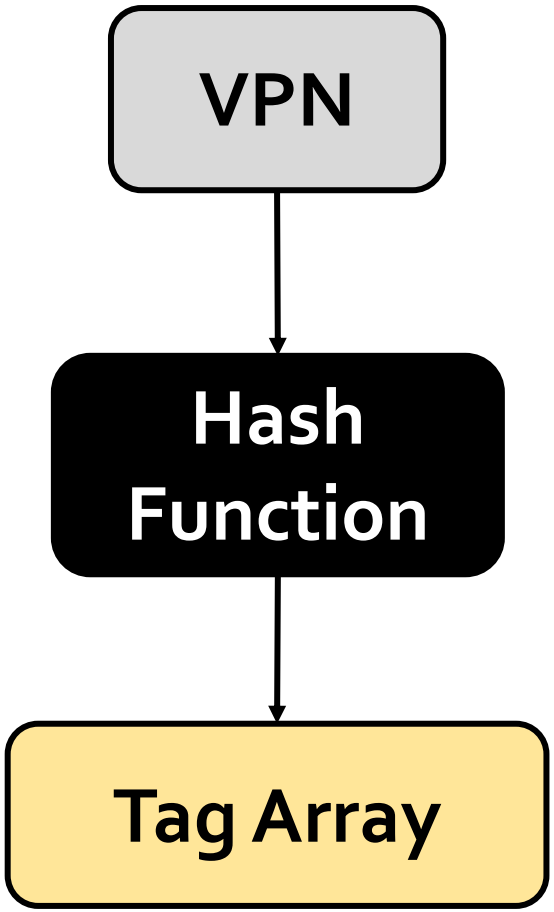
86 cycles on average to access the state-of-the-art hash-based PT

OS: Allocation in RestSeg



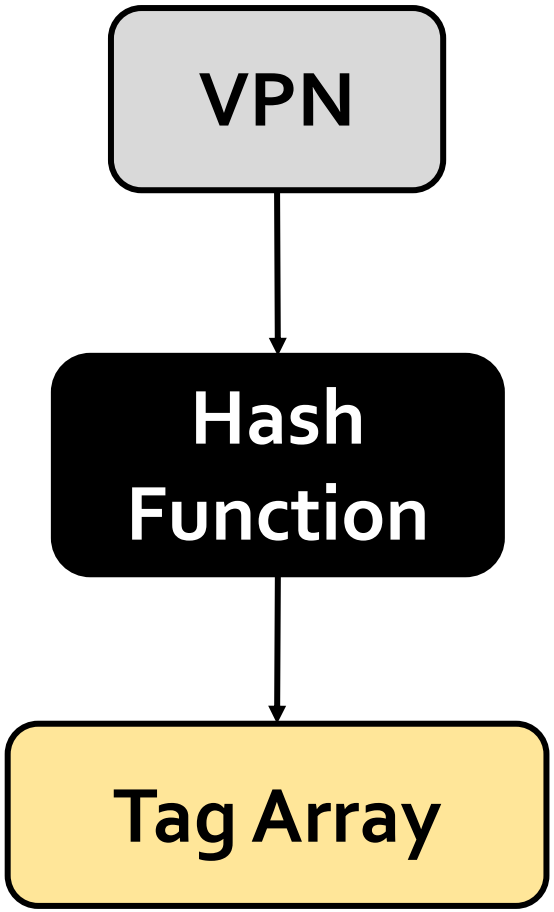
There is a free way!
Store data there

OS: Allocation in RestSeg



Discover if there is a free way in the set

OS: Replacement in RestSeg

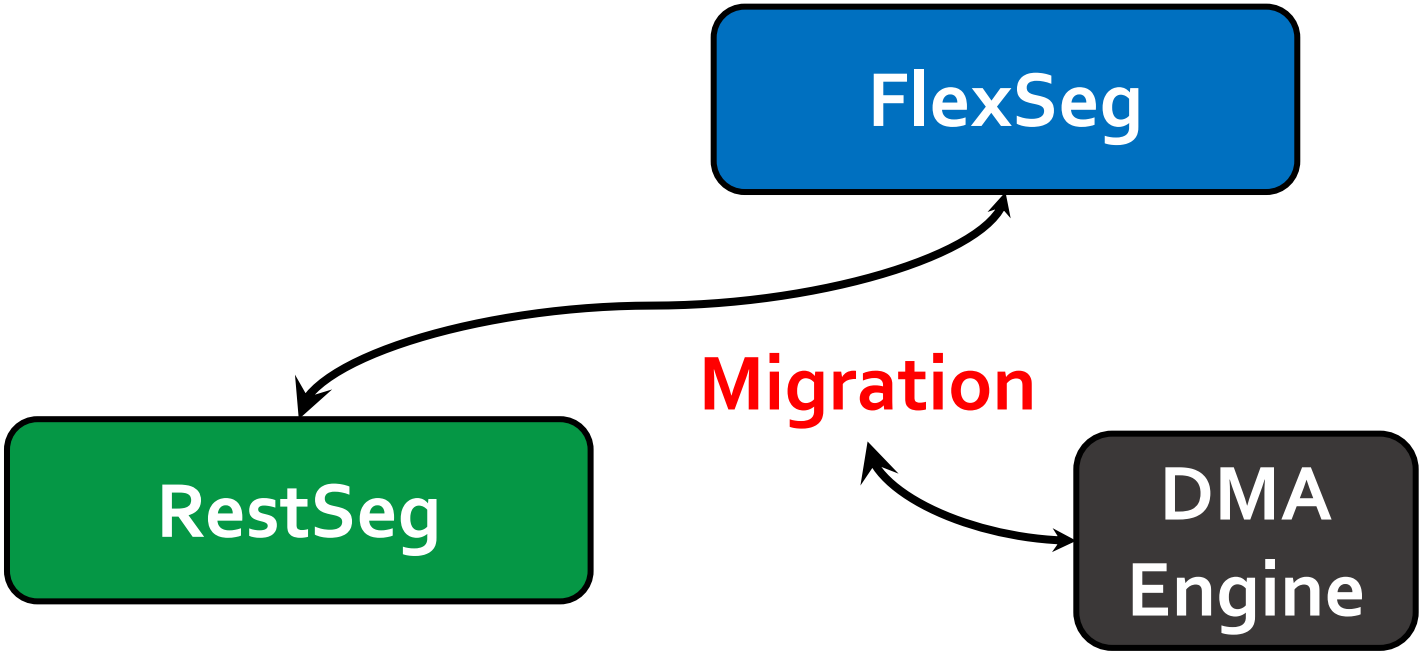


No way is free!

If no page-fault, evict a page!

OS: Migration to/from RestSeg

DMA engine is responsible for migrating data between RestSegs and FlexSegs



Architectural Support for Utopia

New hardware components to support Utopia:

- Specialized hardware circuitry to perform the tag matching and the set filtering
  - Avoid expensive software-based accesses to translation structures
- Small caches for the the Tag Array and Set Filter
  - Accesses to Permissions Filter and Tag Array may exhibit high spatial and temporal locality
- Minor modifications in the address translation pipeline:
  - RSW occurs in parallel with L2 TLB access