MetaSys

A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations

Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos,
F. Nisa Bostancı, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI

ETH Zürich

Carnegie Mellon University

UNIVERSITY OF TORONTO

Executive Summary

Motivation: Hardware-software cooperative (cross-layer) techniques improve performance, quality of service, and security

Problem: Cross-layer techniques are challenging to implement and evaluate in real hardware because they require modifications across the stack

Our Goal is twofold:

1) Enable rapid implementation and evaluation of cross-layer techniques in real hardware

2) Quantify the overheads associated with a general metadata management system

Key Idea:

Develop a metadata management system (**MetaSys**) with hardware and software components that are reusable across cross-layer techniques to minimize programmer effort

Prototype: Xilinx Zedboard prototype using RISC-V Rocket Chip:

Low on-chip storage (0.2%) and memory storage (0.2%) overhead metadata management

Evaluation: Graph prefetching and memory protection case studies, extensive overhead characterization

- MetaSys-based graph prefetcher performs similar to state-of-the-art specialized prefetcher
- Low-overhead bounds-checking (14%) and return address protection (1.2%)
- Single general metadata management system scales well

MetaSys Source Code: https://github.com/CMU-SAFARI/MetaSys

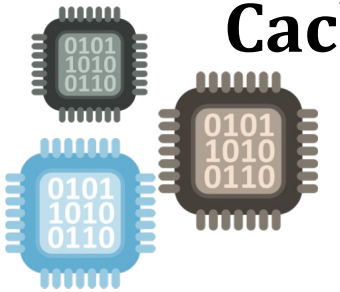
Outline

- **Background**
  - **Cross-Layer Techniques**
  - **Evaluating Cross-Layer Techniques**
- MetaSys
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

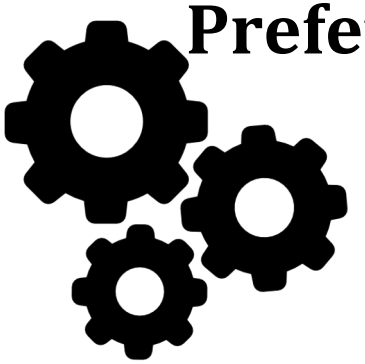
Hardware Performance Optimizations

Today

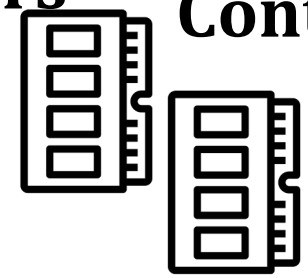
Optimize for performance by designing hardware that infers and predicts program behavior



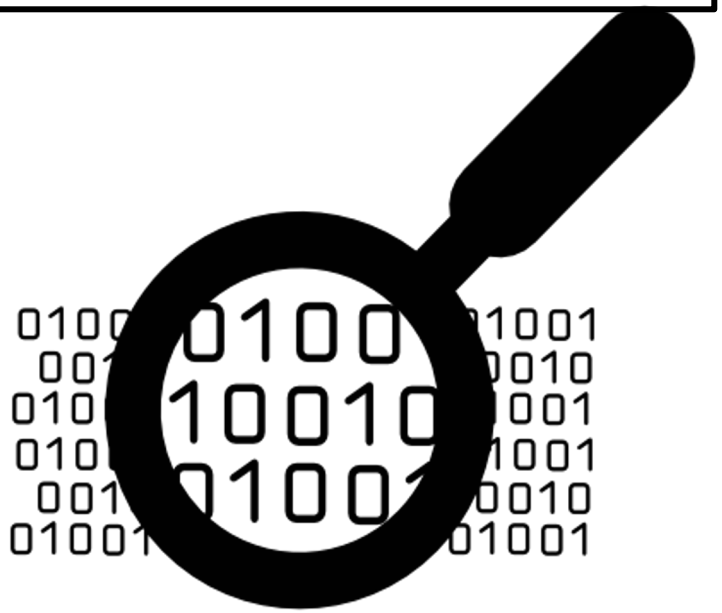
Caches



Prefetchers



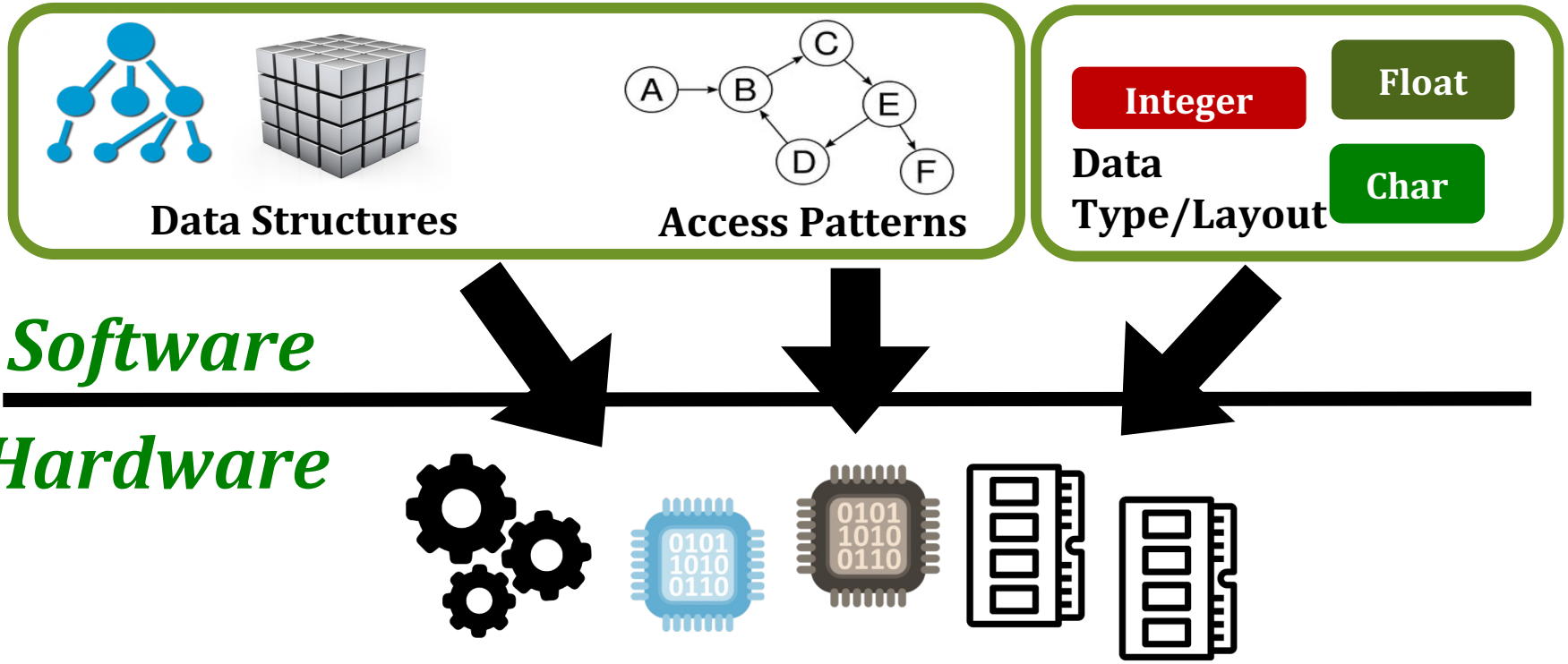
Memory Controllers



Cross-Layer Techniques

Future: Cross-Layer

Software can provide information (metadata) that the hardware is trying to infer



Example: Locality Descriptor (I)

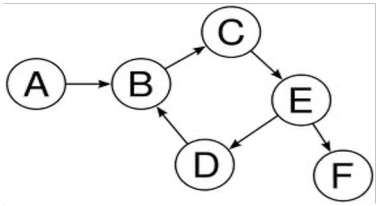
Locality Descriptor [Vijaykumar+, ISCA'18a]

1. Express data locality (from software)
2. Exploit data locality (in hardware) in GPUs

The programmer or the compiler describes key semantics using a software interface

```
cudaMalloc(sm_mappings, size);
LocalityDescriptor ldesc(sm_mappings, size, INTER-THREAD, tile,
```

Data Structure

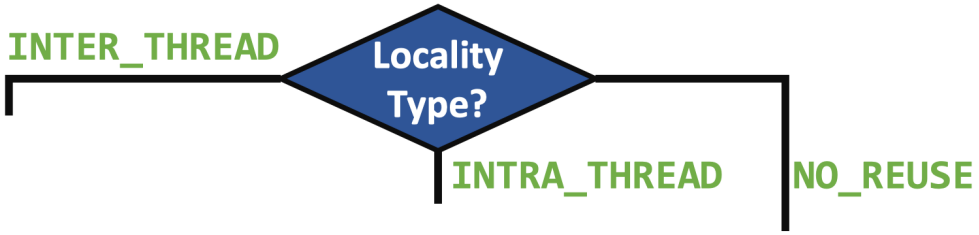


Float

Char

Metadata

Example: Locality Descriptor (II)



Hardware optimizations
leverage key program semantics
to improve performance significantly (e.g., >50%)

Benefits of Cross-Layer Techniques

- Performance optimizations
- Security enhancements
- Quality of service improvements

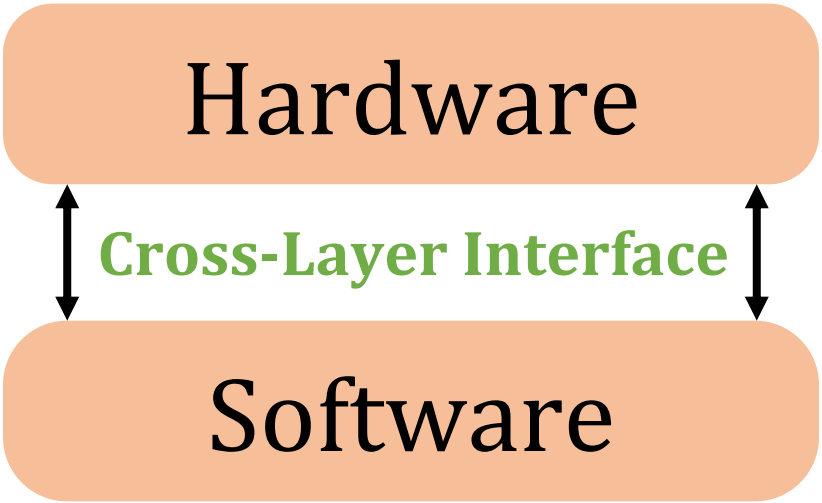
Hardware

- Better programmability
- Better portability

Software

Example prior work

[Vijaykumar+, ISCA'18a]
[Vijaykumar+, ISCA'18b]
[Koo+, ISCA'17]
[Yu+, CARVV'17]
[Mukkara+, ASPLOS'16]
[Ma+, ASPLOS'15]
...

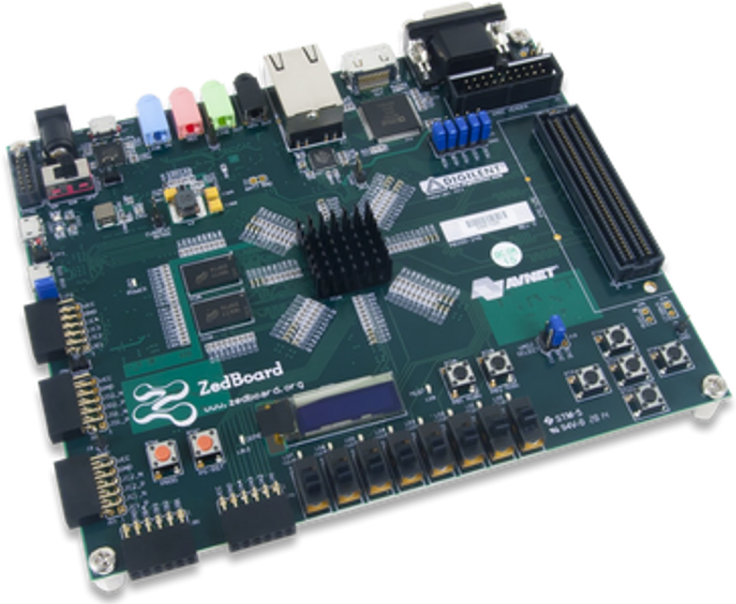


Cross-Layer Evaluation Infrastructure

Evaluating cross-layer techniques is non-trivial

Cycle-accurate simulators

FPGA prototypes

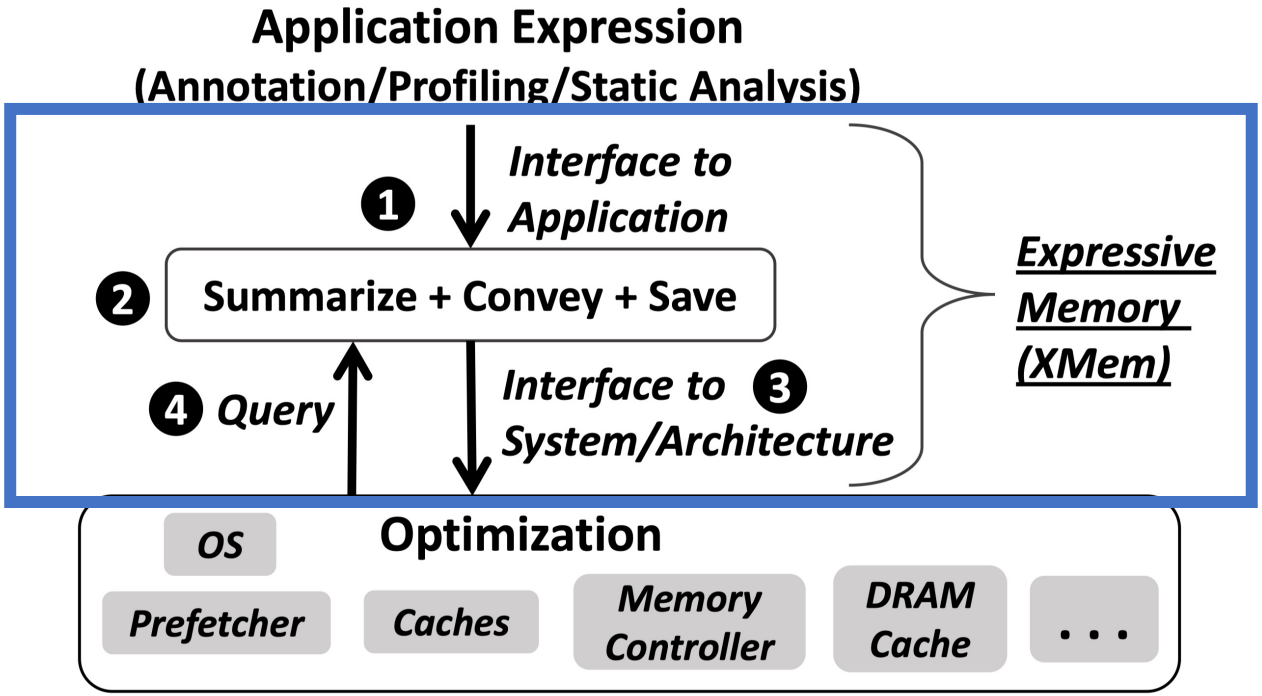


Need a new infrastructure to evaluate a new cross-layer technique

- Difficult given complexity (changes across the stack)

General Metadata Management System

A single general system can support multiple techniques



Key Benefit: Amortize the cost to implement each new technique

[Vijaykumar+, ISCA'18b]

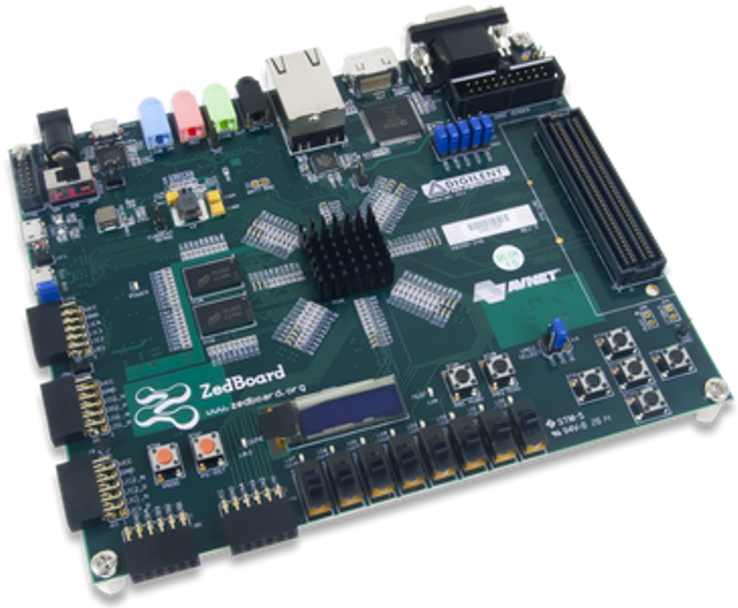
Outline

- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- **MetaSys**
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

MetaSys: Overview

Goal: Enable rapid implementation and evaluation of cross-layer techniques in real hardware

MetaSys: Open-source infrastructure to evaluate cross-layer techniques end-to-end



CMU-SAFARI / MetaSys Public

Notifications | Fork 3 | Star 3

Code | Issues | Pull requests | Actions | Projects | Security | Insights

main | Go to file | Code

olgunataberk Update README.md ... on Jul 9, 2021 12

common	Initial commit	last year
riscv-tools	Add tools directory	last year
rocket-chip	Initial commit	last year
testchipip	Initial commit	last year
zedboard	Initial commit	last year
LICENSE	Initial commit	last year
README.md	Update README.md	last year
metasys_rea...	Update metasys_readme.md	last year

README.md

About

Metasys is the first open-source FPGA-based infrastructure with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer software/hardware cooperative techniques techniques in real hardware. Described in our pre-print: https://arxiv.org/abs/2105.08123

Readme
View license
3 stars
4 watching
3 forks

MetaSys Source Code: https://github.com/CMU-SAFARI/MetaSys

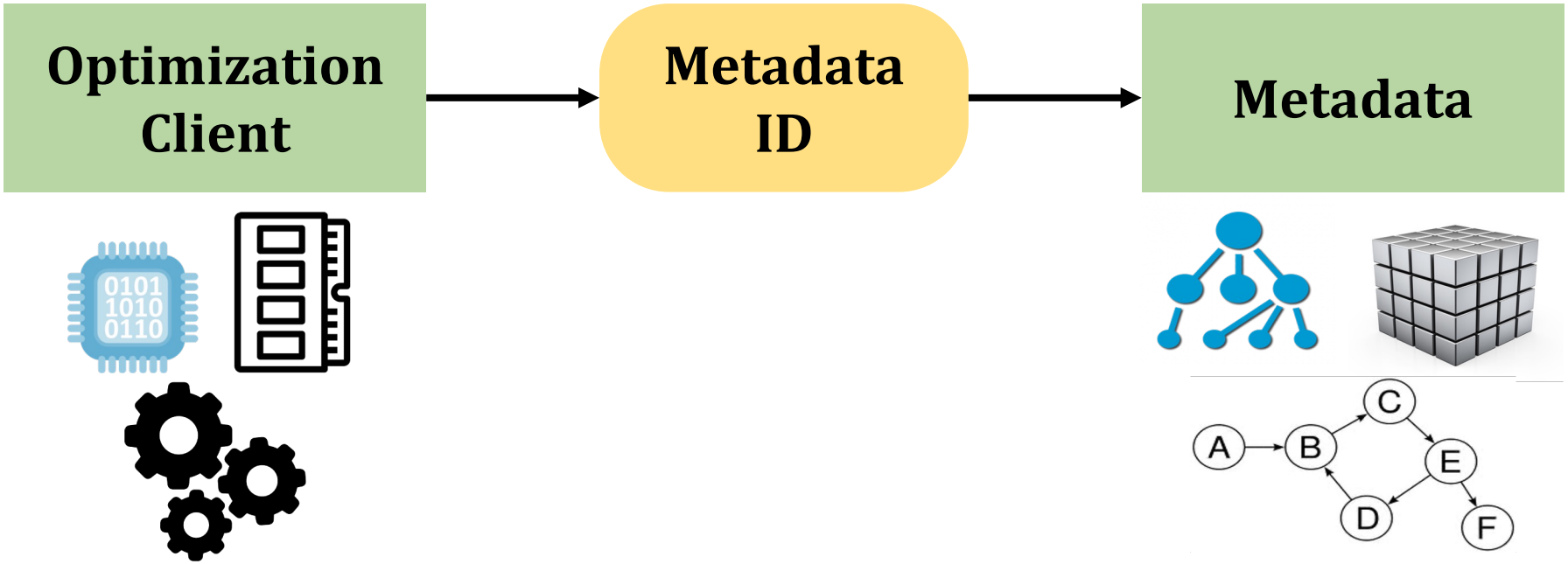
MetaSys: Components

1

Tagged memory-based metadata management
Efficient metadata querying in hardware

MetaSys: Tagged Memory (I)

Tag memory addresses with metadata IDs

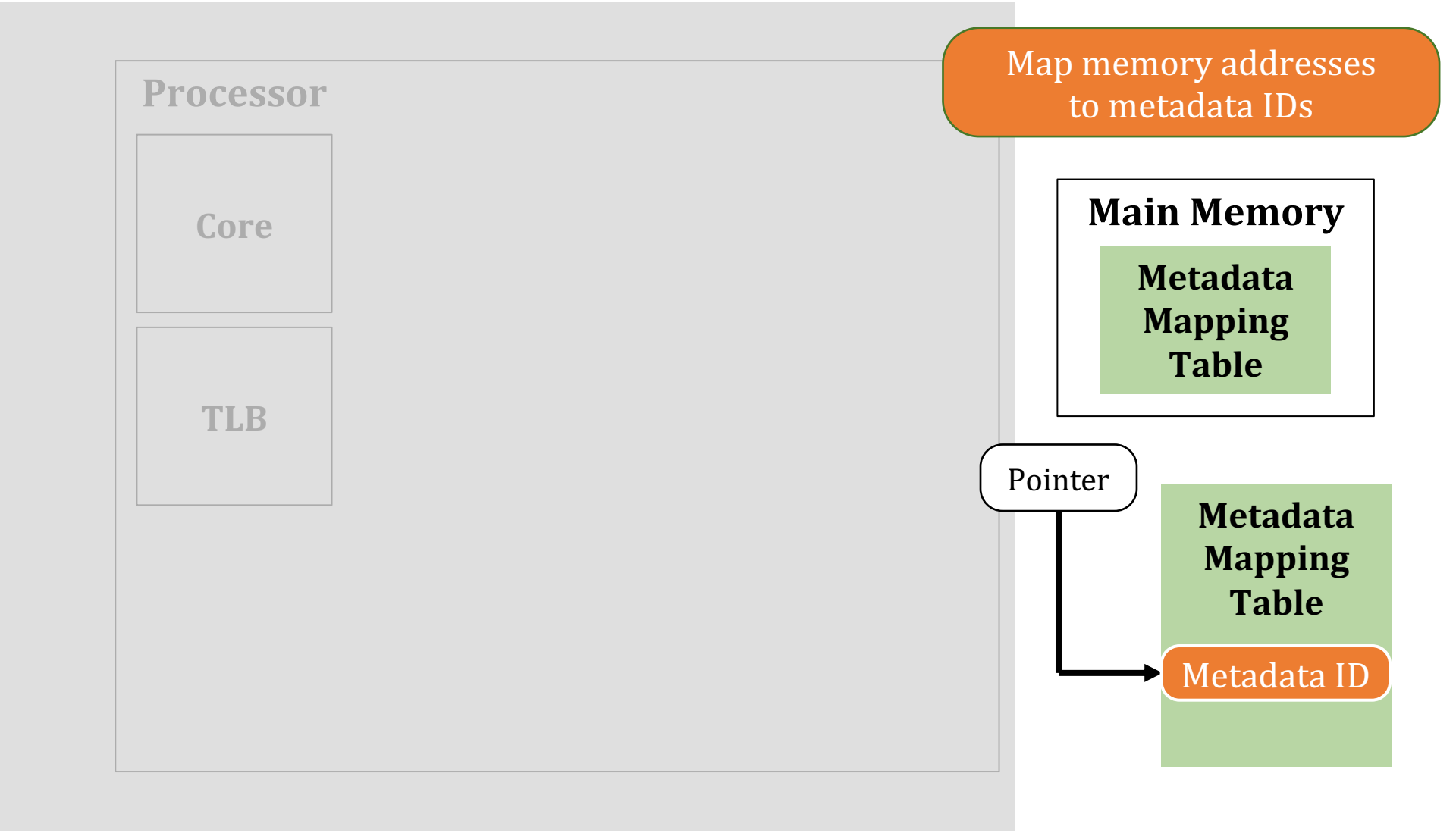


Example

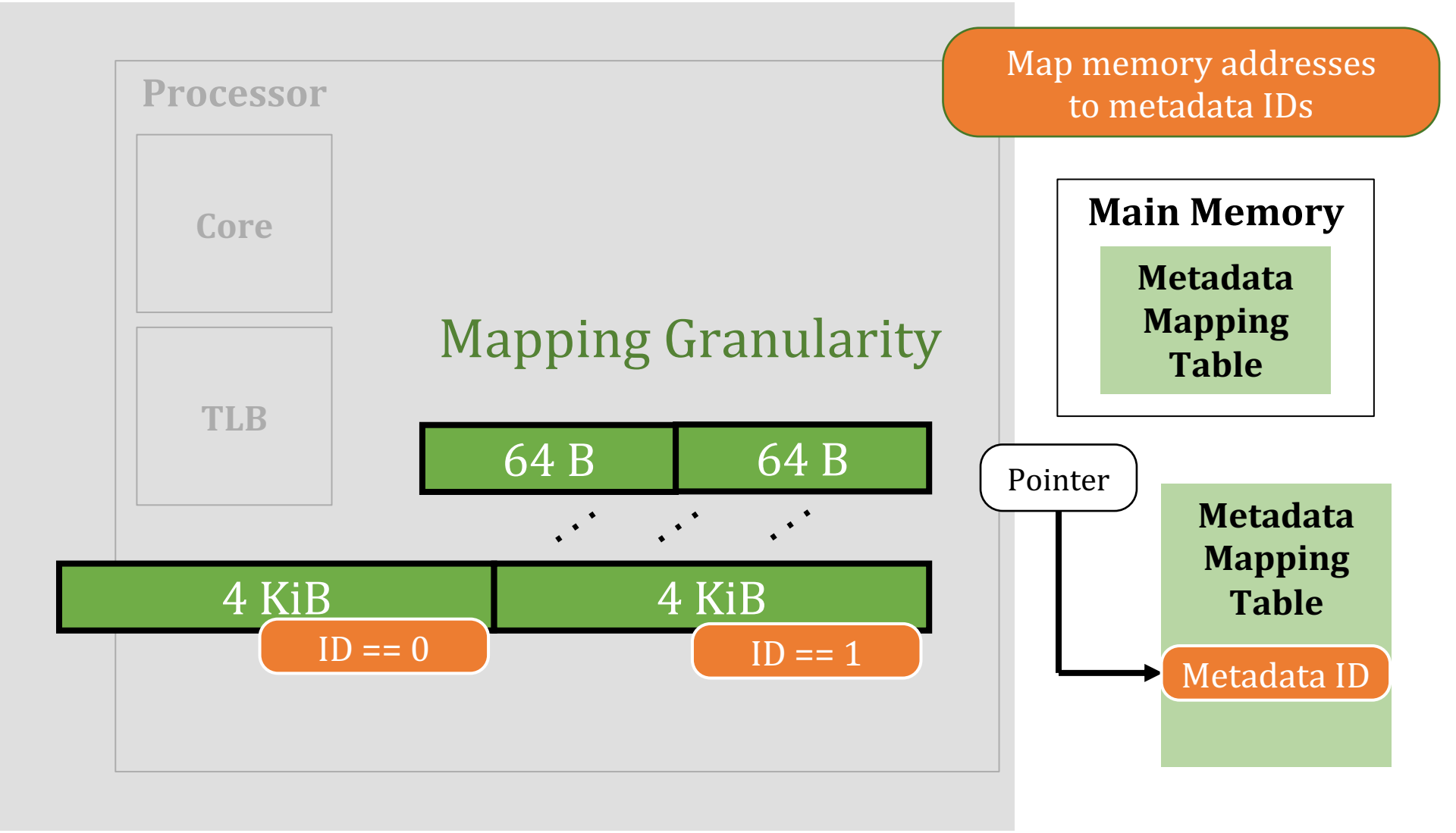
Is <address> accessed frequently? → metadata ID → Metadata Store: YES

MetaSys: Tagged Memory (II)

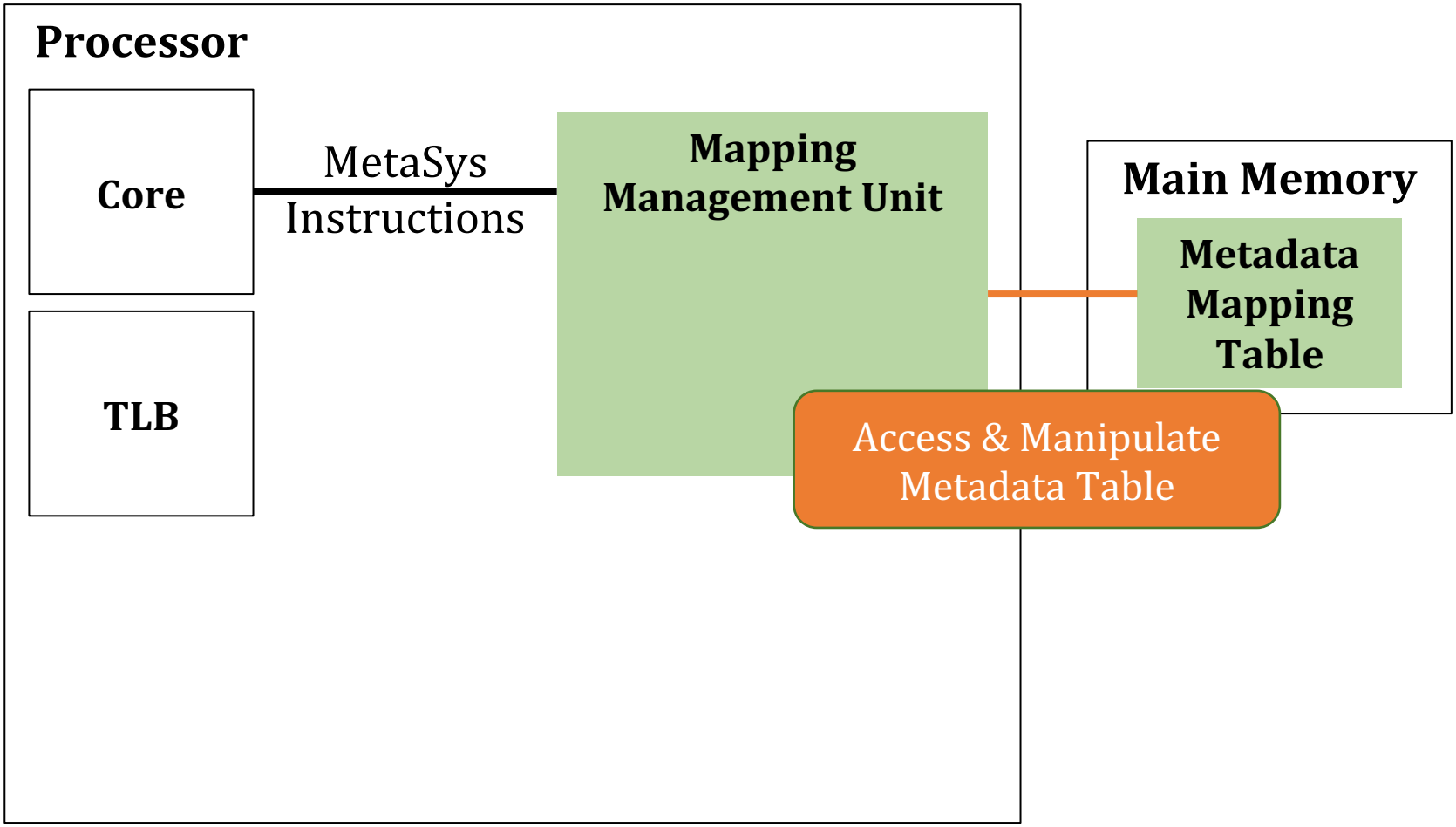
MetaSys: Tagged Memory (II)



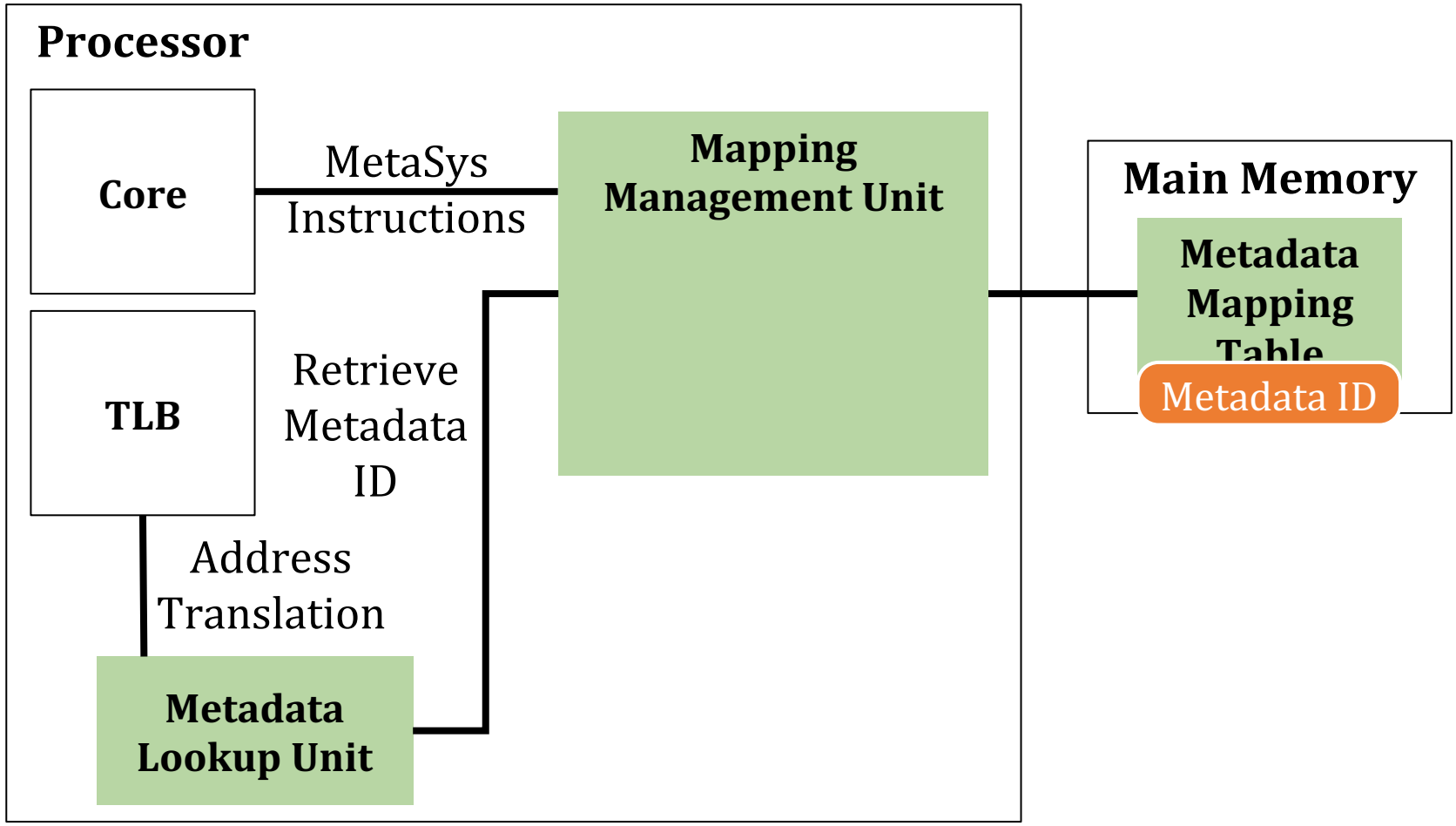
MetaSys: Tagged Memory (II)



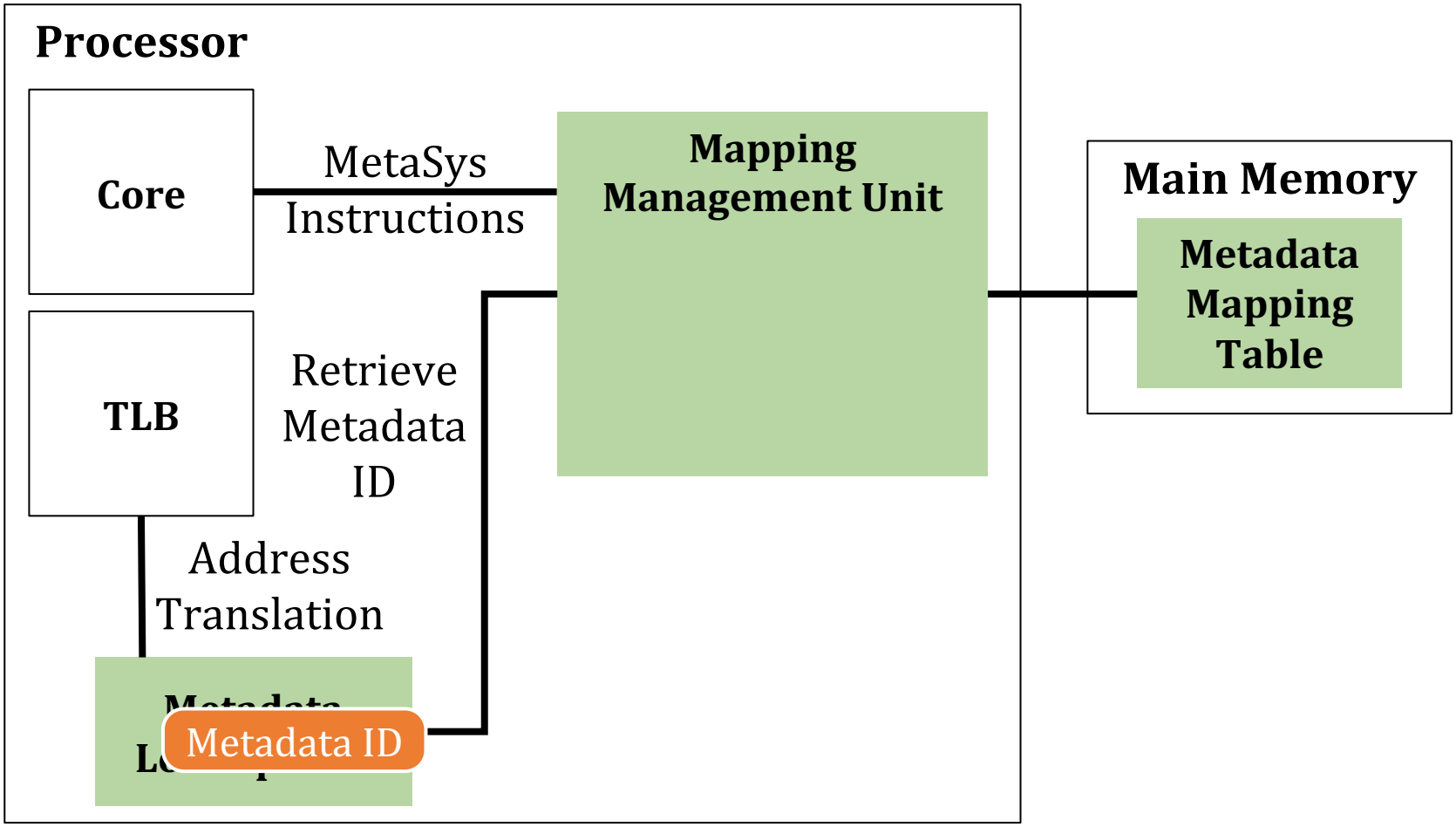
MetaSys: Tagged Memory (II)



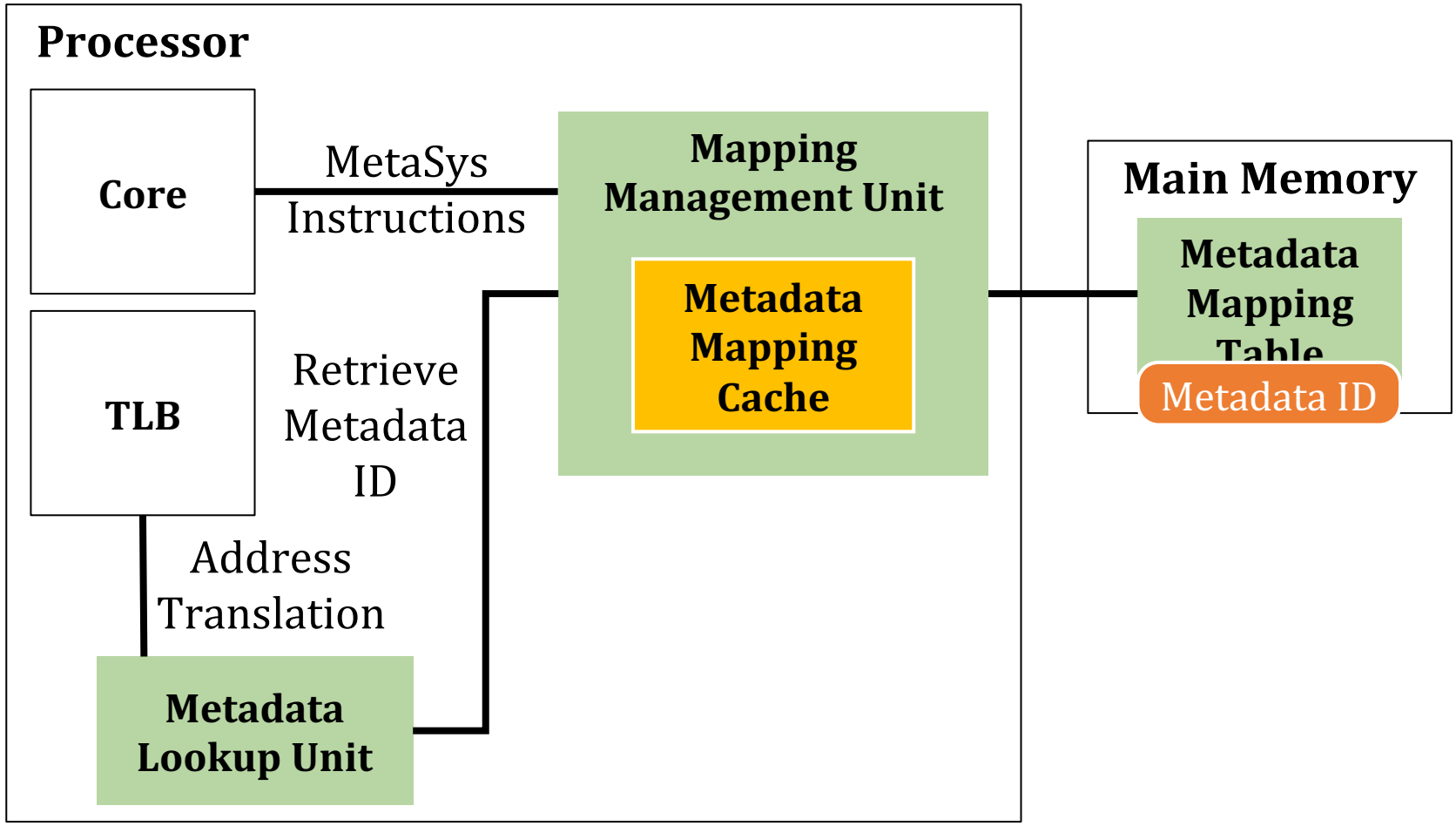
MetaSys: Tagged Memory (II)



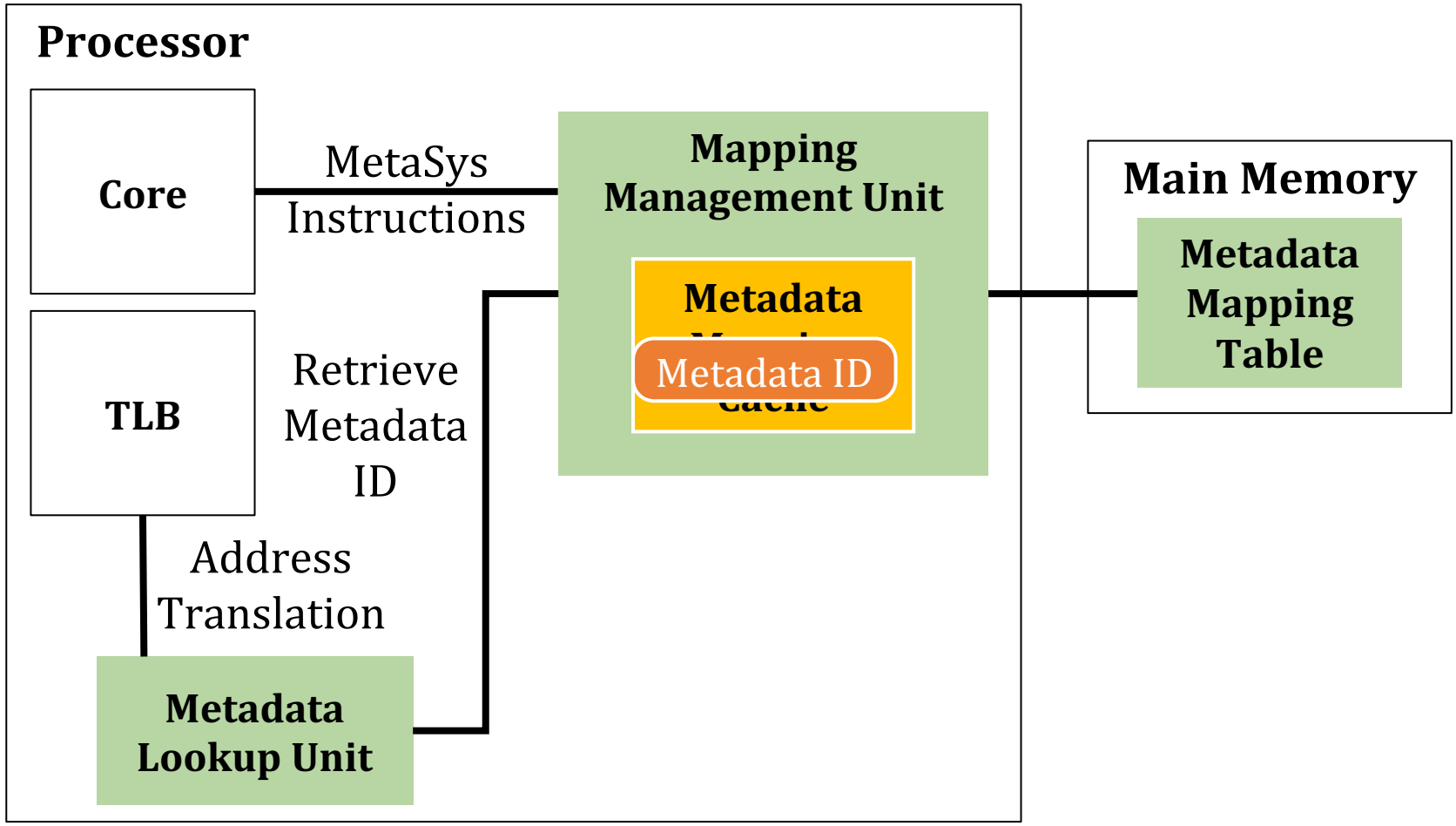
MetaSys: Tagged Memory (II)



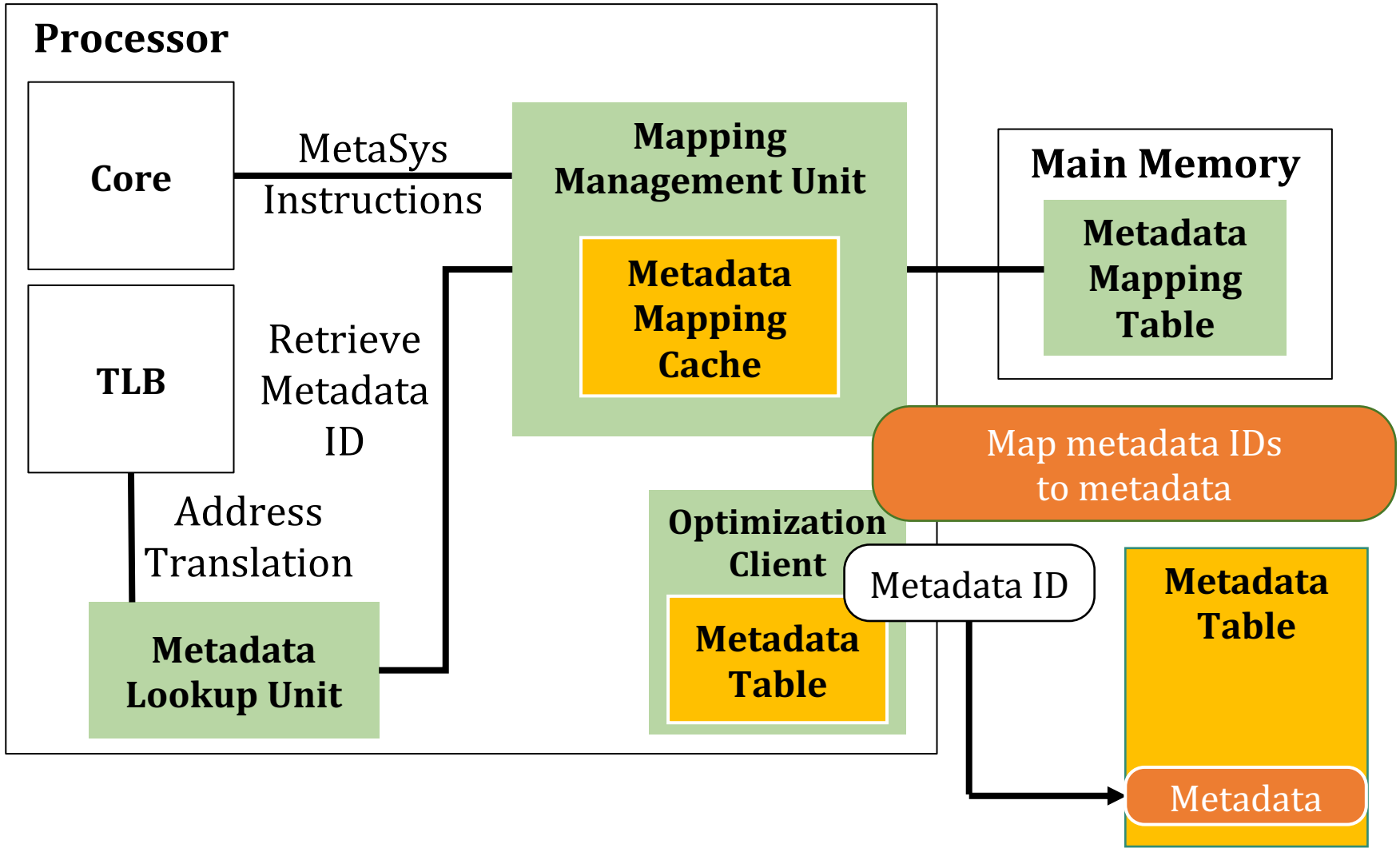
MetaSys: Tagged Memory (II)



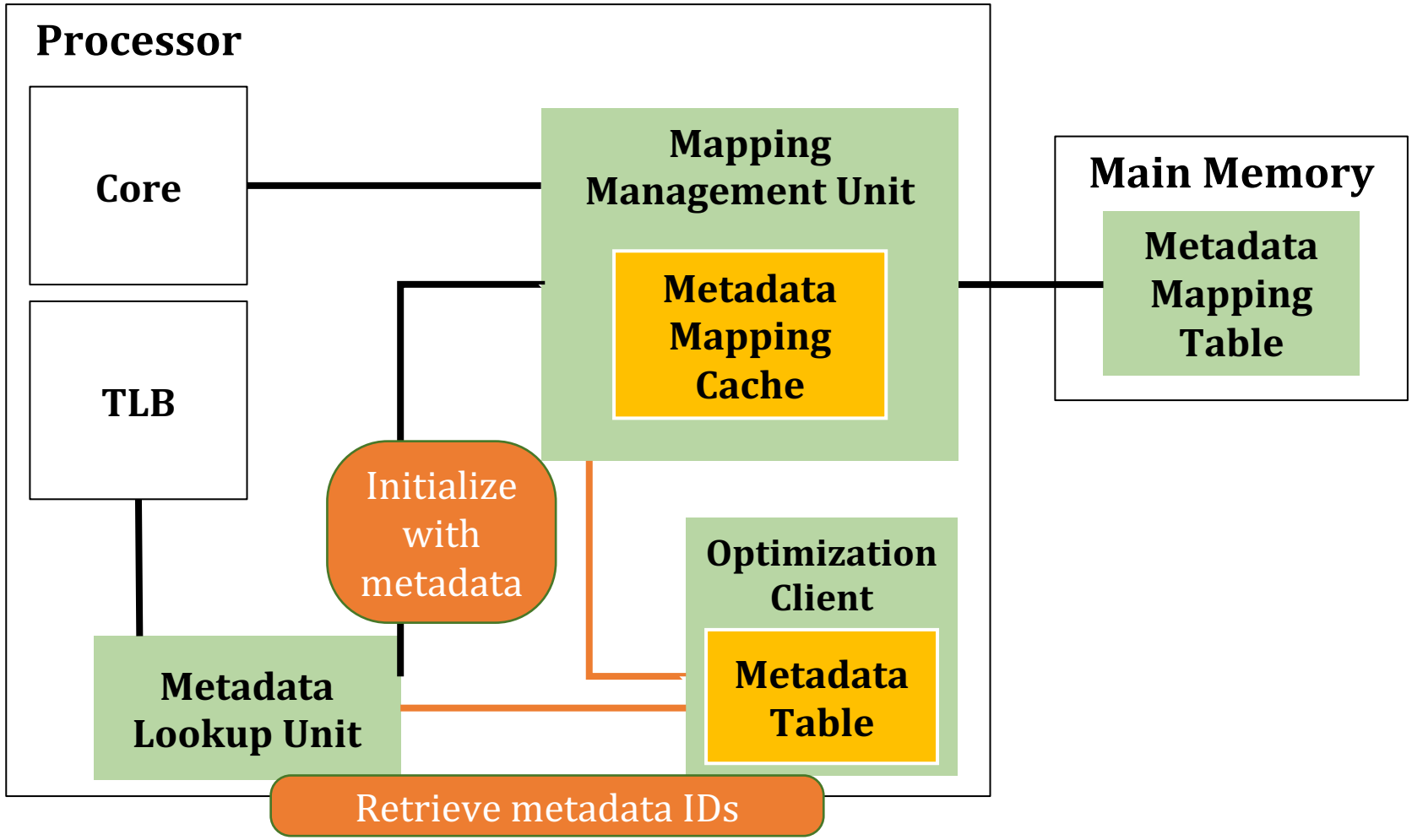
MetaSys: Tagged Memory (II)



MetaSys: Tagged Memory (II)



MetaSys: Tagged Memory (II)

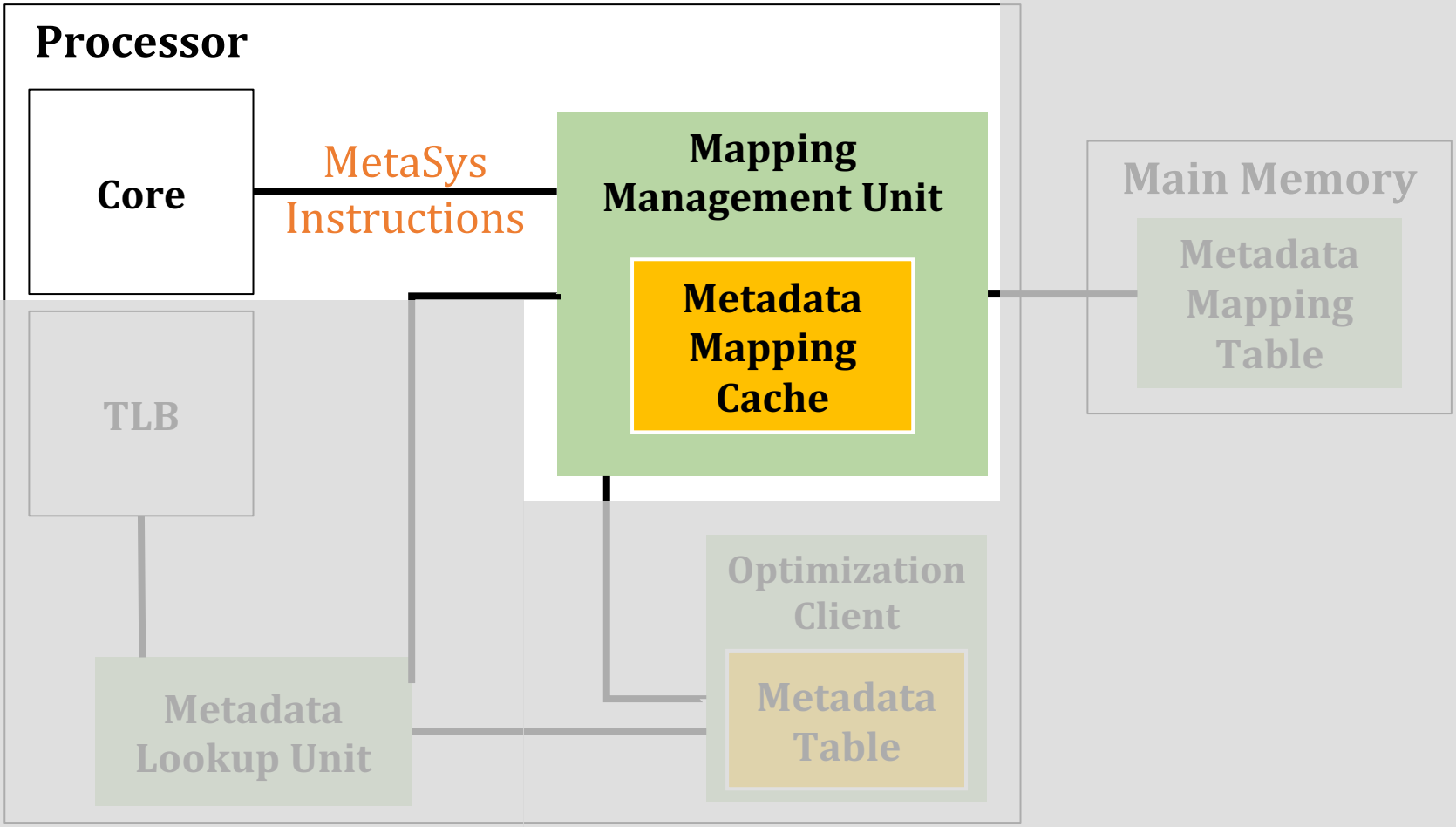


MetaSys: Components

1. **Tagged memory**-based metadata management
   Efficient metadata querying in hardware

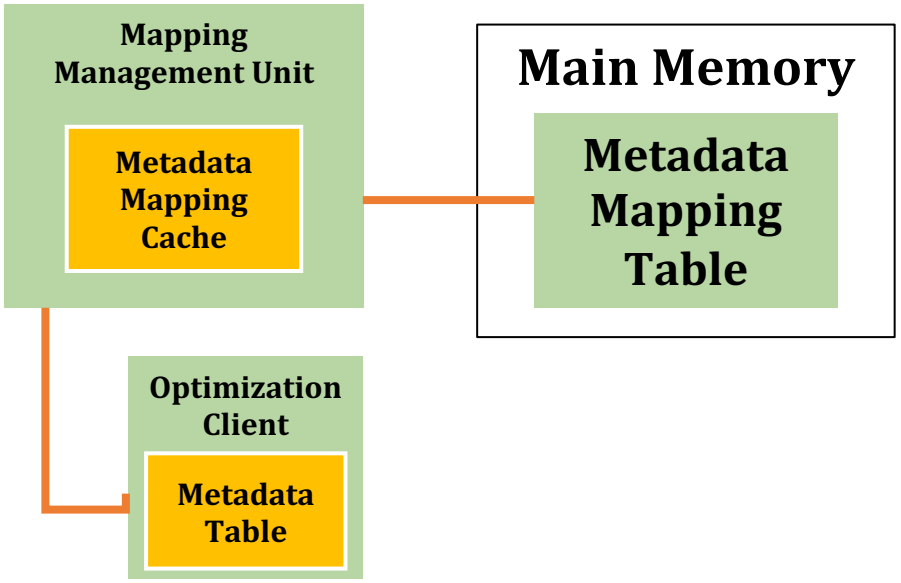
2. Rich **cross-layer interface**
   Communicate metadata from SW to HW

MetaSys: Cross-Layer Interface



MetaSys: Cross-Layer Interface

MetaSys Operator	MetaSys ISA Instructions
CREATE	CREATEClientID, TagID, Metadata
(UN)MAP	(UN)MAP TagID, start_addr, size (UN)MAP2D TagID, start_addr, lenX, sizeX, sizeY (UN)MAP3D TagID, start_addr, lenX, lenY, sizeX, sizeY, sizeZ;



MetaSys: Cross-Layer Interface

MetaSys Operator	MetaSys ISA Instructions
CREATE	CREATEClientID, TagID, Metadata
(UN)MAP	(UN)MAP TagID, start_addr, size (UN)MAP2D TagID, start_addr, lenX, sizeX, sizeY (UN)MAP3D TagID, start_addr, lenX, lenY, sizeX, sizeY, sizeZ;

RESEARCH-ARTICLE OPEN ACCESS

MetaSys: A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations

Authors: Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, + 1

Authors Info & Claims

ACM Transactions on Architecture and Code Optimization, Volume 19, Issue 2 • June 2022 • Article No.: 26, pp 1–29 • https://doi.org/10.1145/3505250

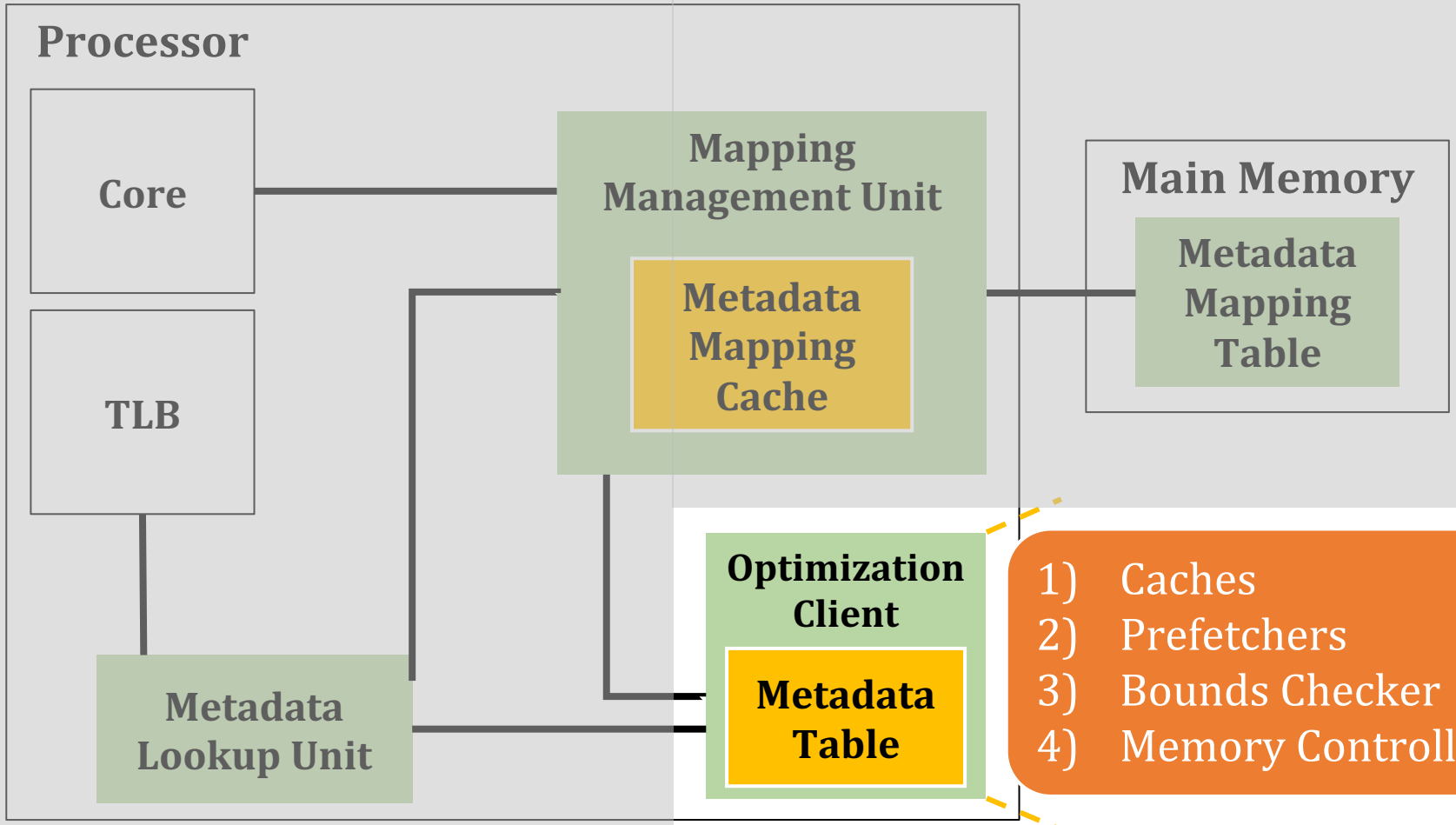
MetaSys: Components

1. **Tagged memory**-based metadata management
   Efficient metadata querying in hardware

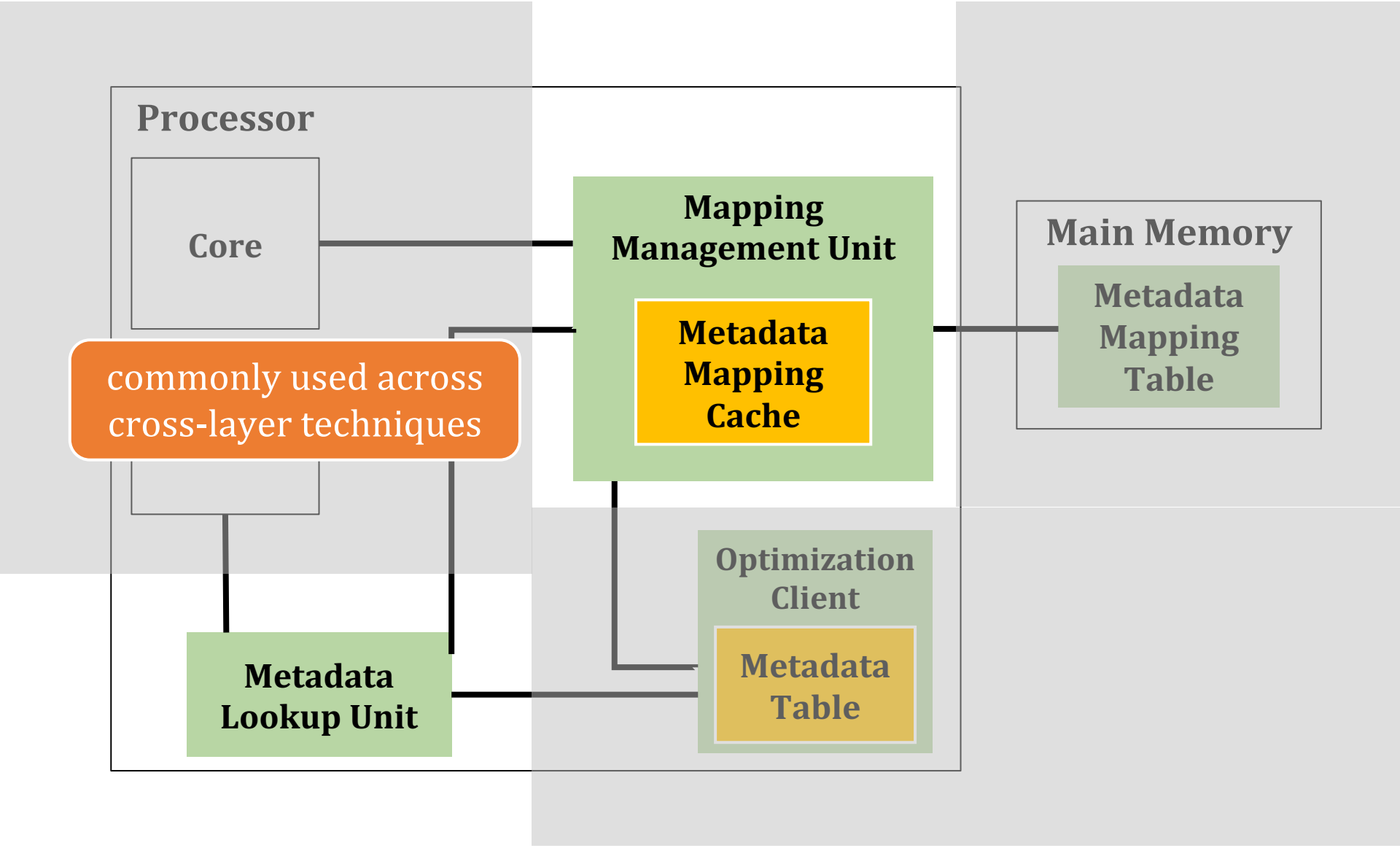
2. Rich **cross-layer interface**
   Communicate metadata from SW to HW

3. Flexible **hardware** and **software modules**
   Facilitate implementing new HW optimizations

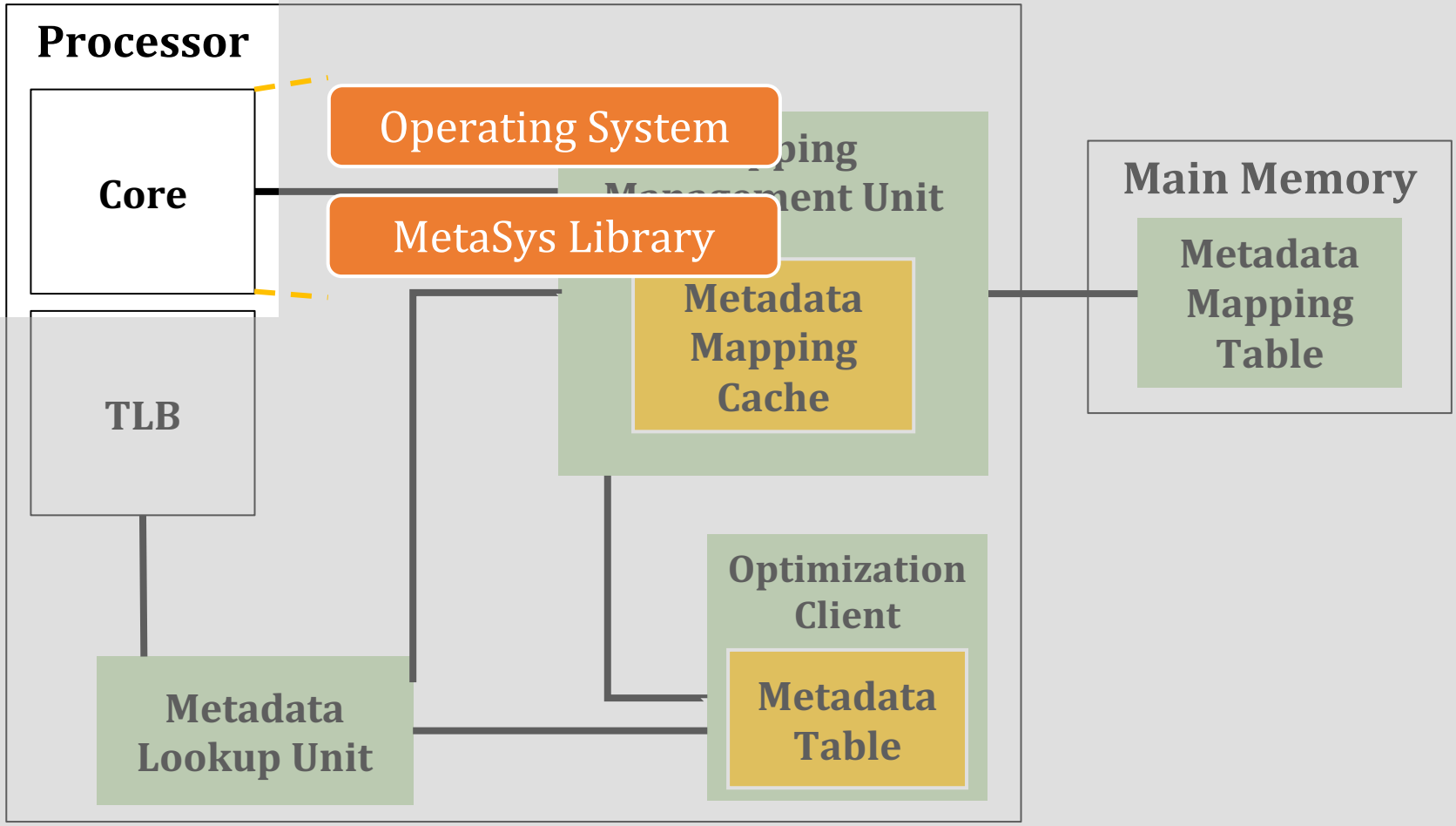
MetaSys: Hardware Modules (I)



MetaSys: Hardware Modules (II)



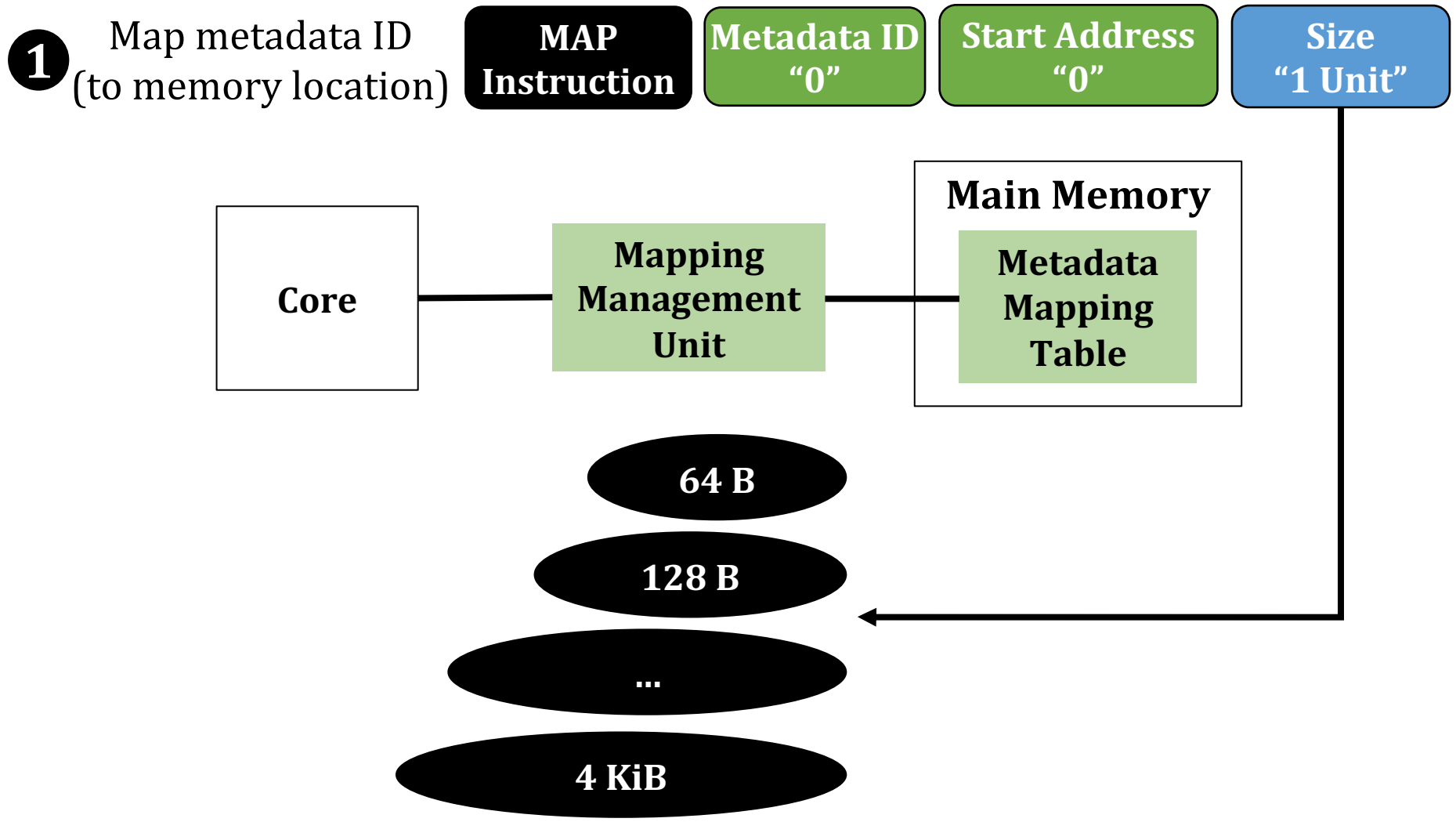
MetaSys: Software Modules



Outline

- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- **MetaSys**
  - Overview
  - Components
  - **Operation**
  - FPGA Prototype
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

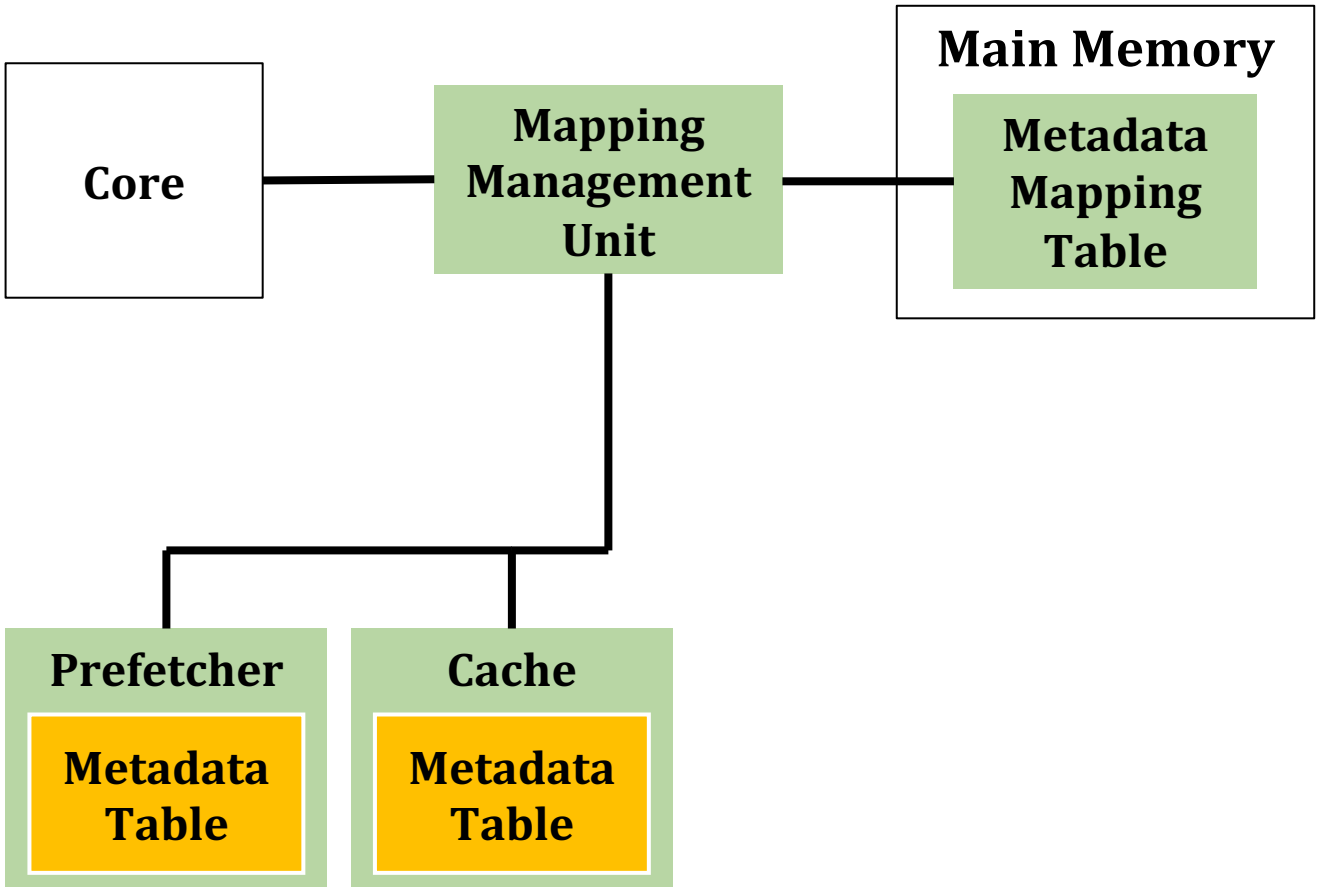
MetaSys: Operation (MAP)



MetaSys: Operation (CREATE)

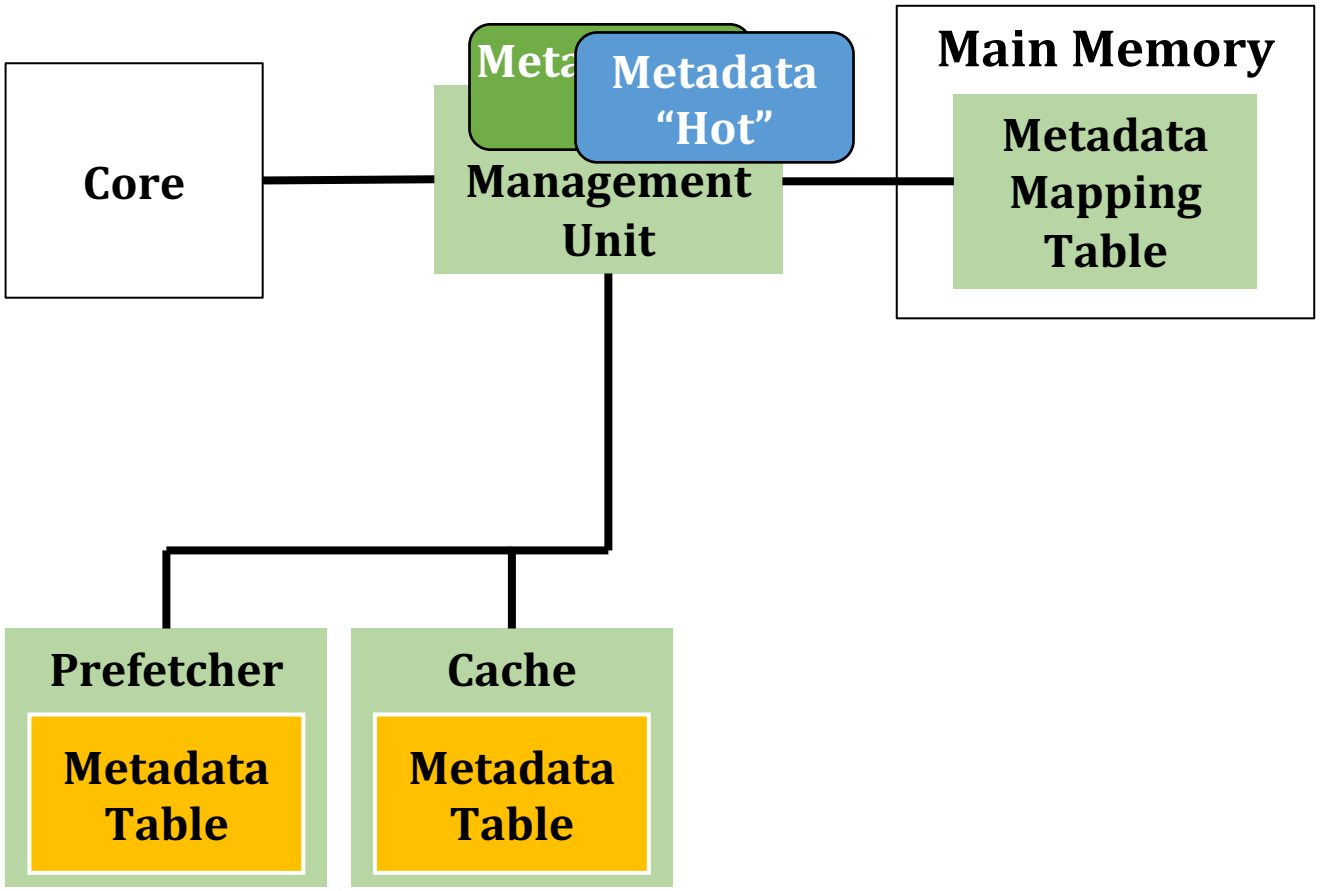
❷ Associate metadata (with metadata ID)

CREATE Instruction | **Client ID "Cache"** | **Metadata ID "0"** | **Metadata "Hot"**

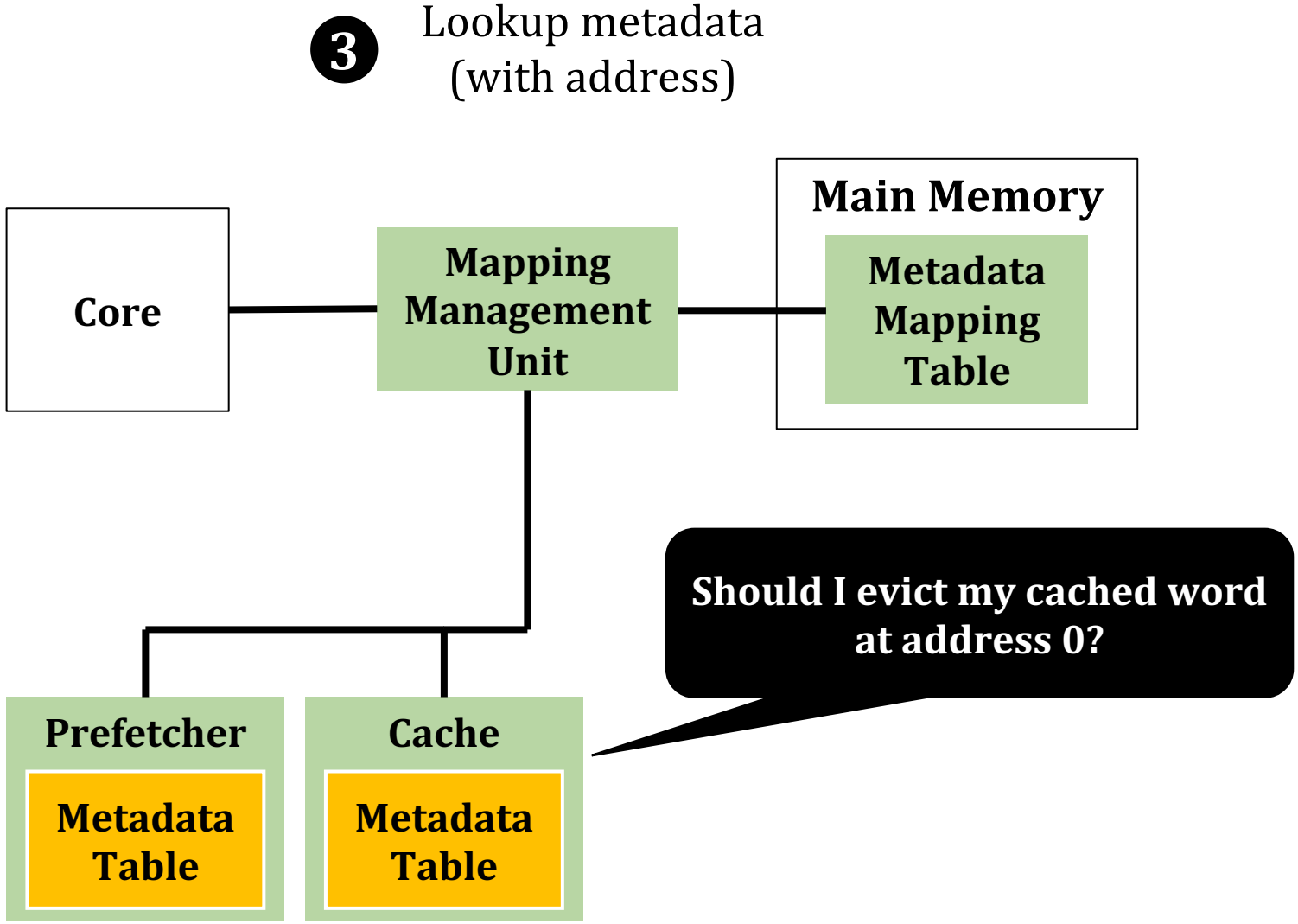


MetaSys: Operation (CREATE)

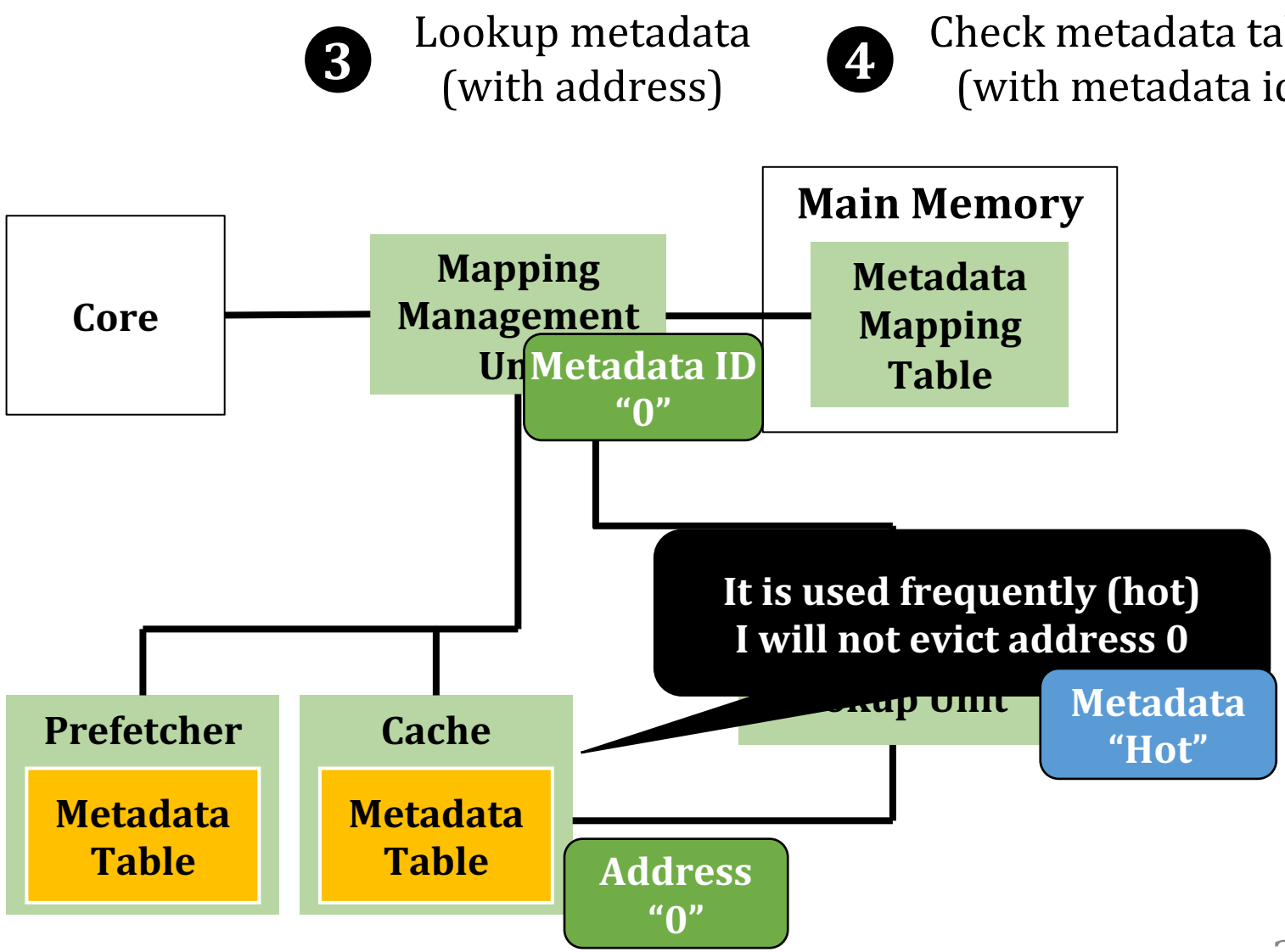
❷ Associate metadata (with metadata ID)



MetaSys: Operation (LOOKUP)



MetaSys: Operation (LOOKUP)

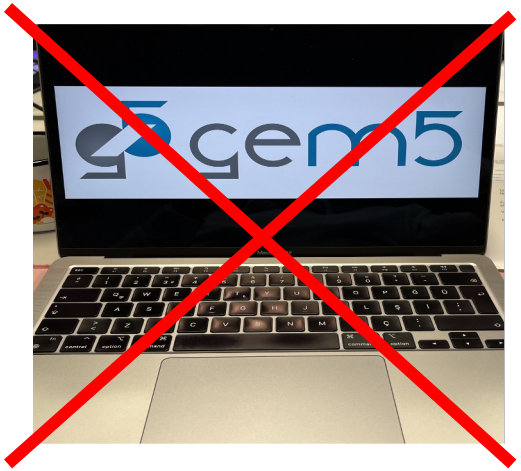


Outline

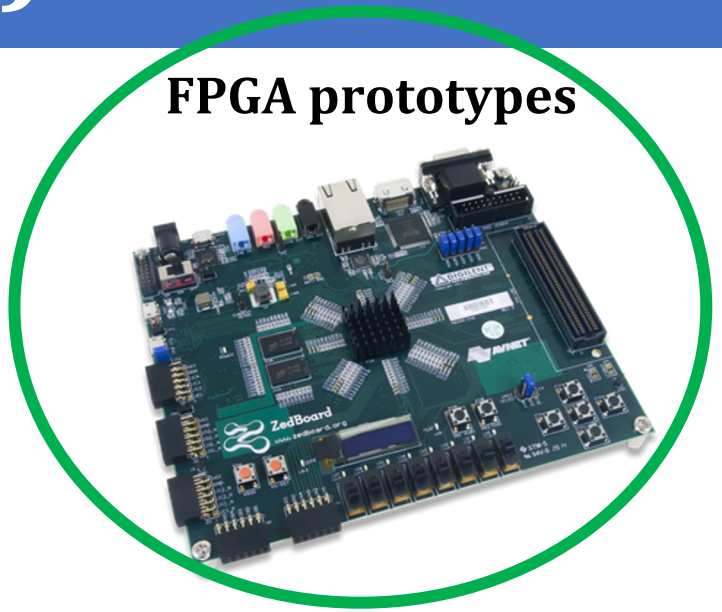
- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- **MetaSys**
  - Overview
  - Components
  - Operation
  - **FPGA Prototype**
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

FPGA Prototype (I)

Cycle-accurate simulators

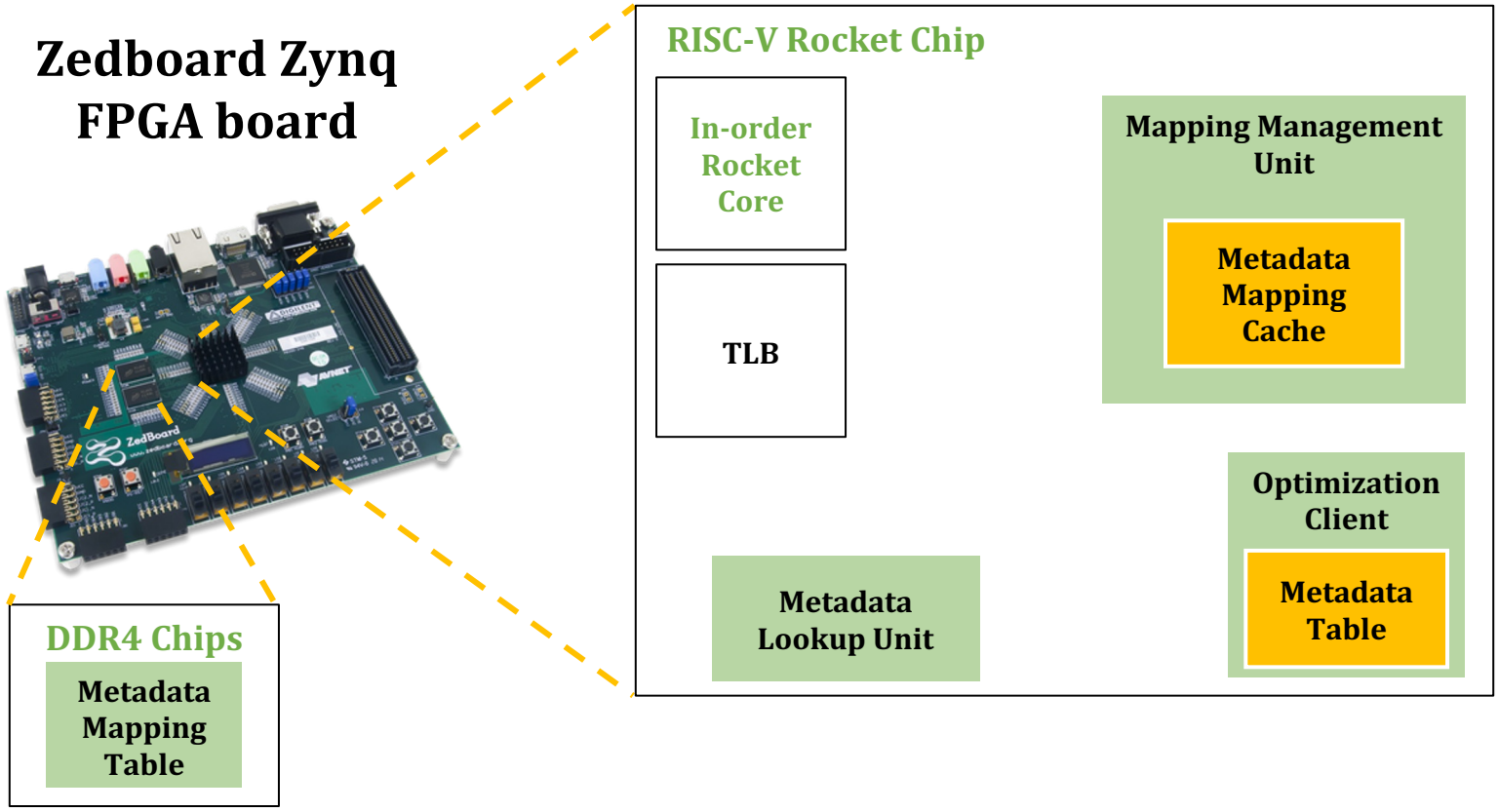


FPGA prototypes



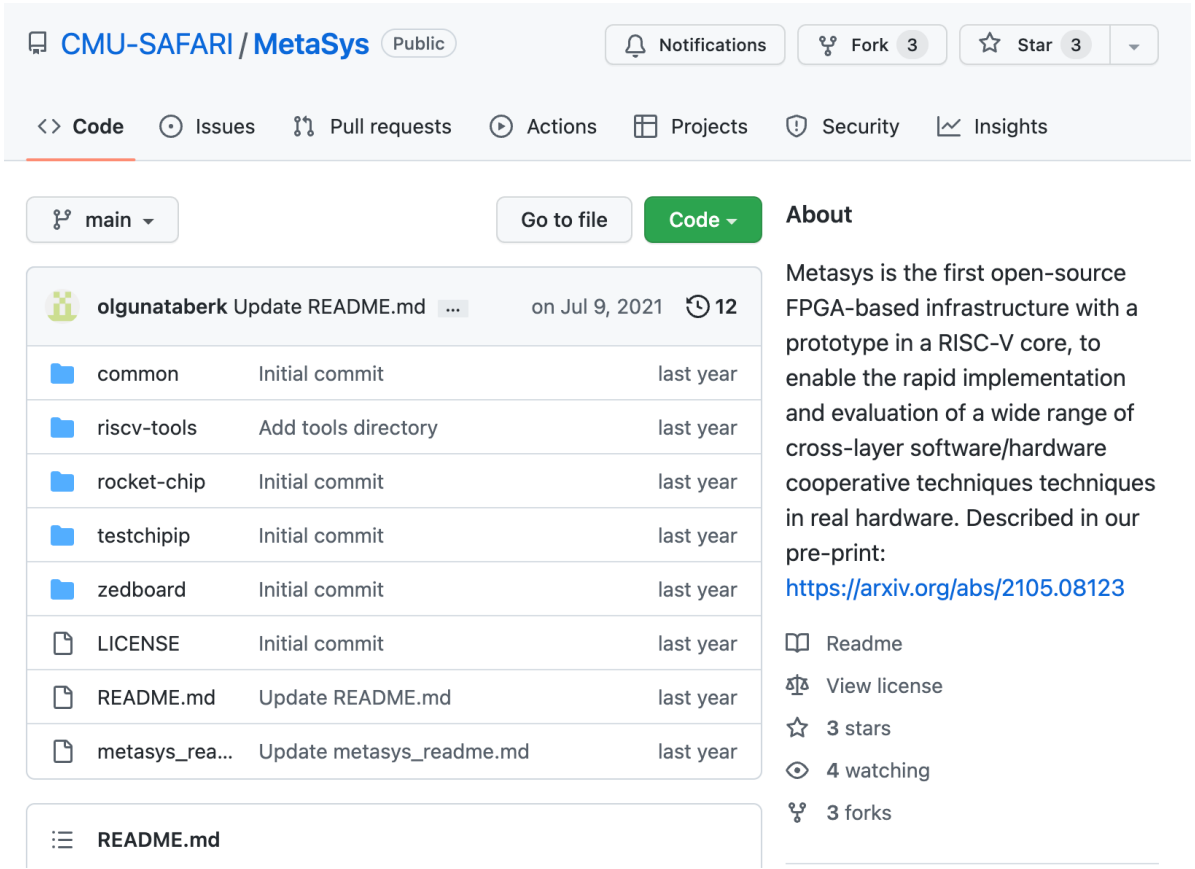
1. Accurately evaluate feasibility of the metadata management system
   - Implement all components (e.g., ports, wires, buffers)
2. Quickly run experiments
   - Run workloads fast compared to simulation
3. Develop RTL as a basis for future work
   - E.g., accurately measure area and power using synthesis tools

FPGA Prototype (II)



MetaSys is Open Source

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



Link to Source Code

Outline

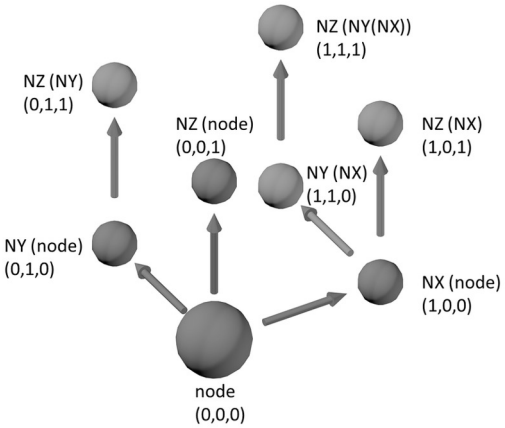
- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- MetaSys
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- **Case Studies**
  - **Prefetching**
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

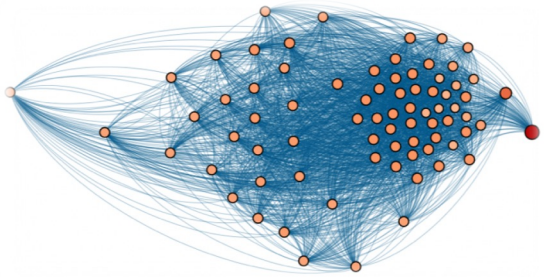
Cross-Layer Prefetching Techniques

Handle challenging access patterns

- Graph processing
- Pointer chasing
- Linear algebra
- …

Example Prior Work
[Ainsworth+, ICS'16]
[Talati+, HPCA'21]





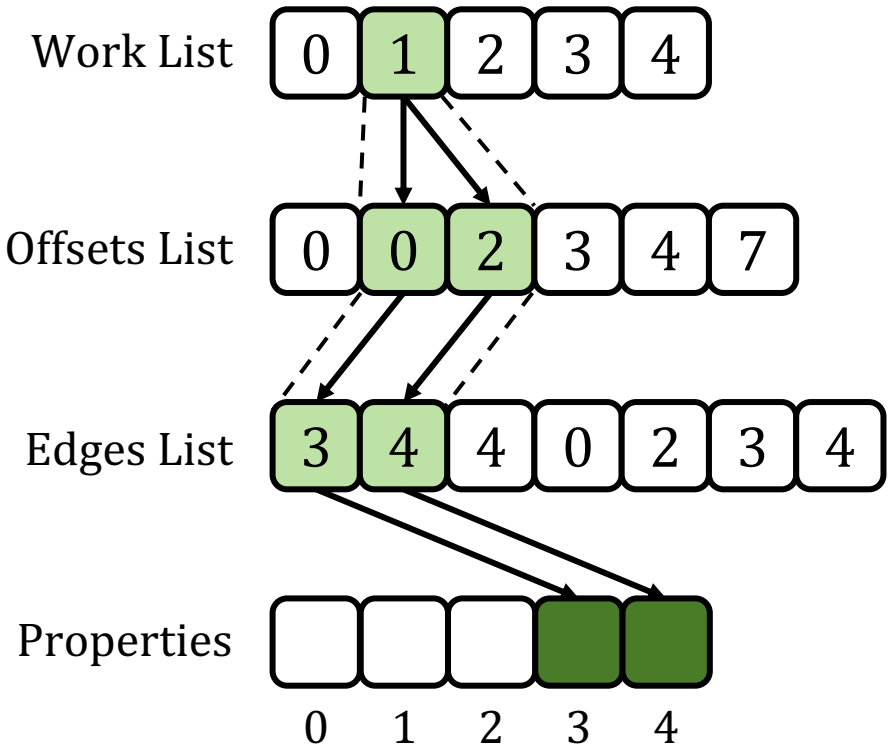
Case Study #1

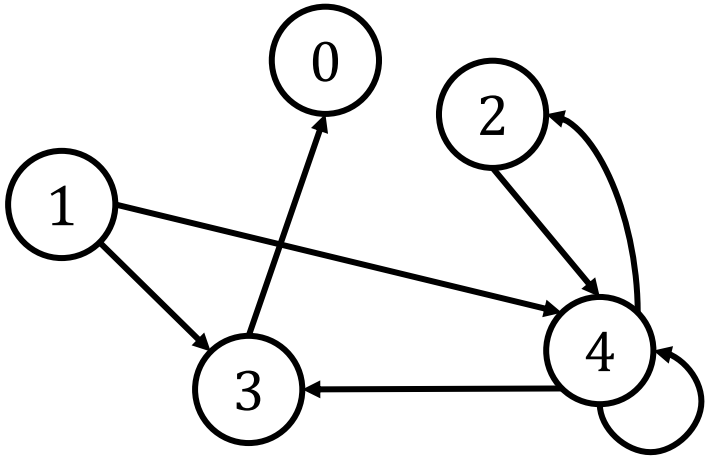
New cross-layer prefetching technique for graph applications

- Leverage graph data structure semantics
  - Instead of relying on program context or memory access history

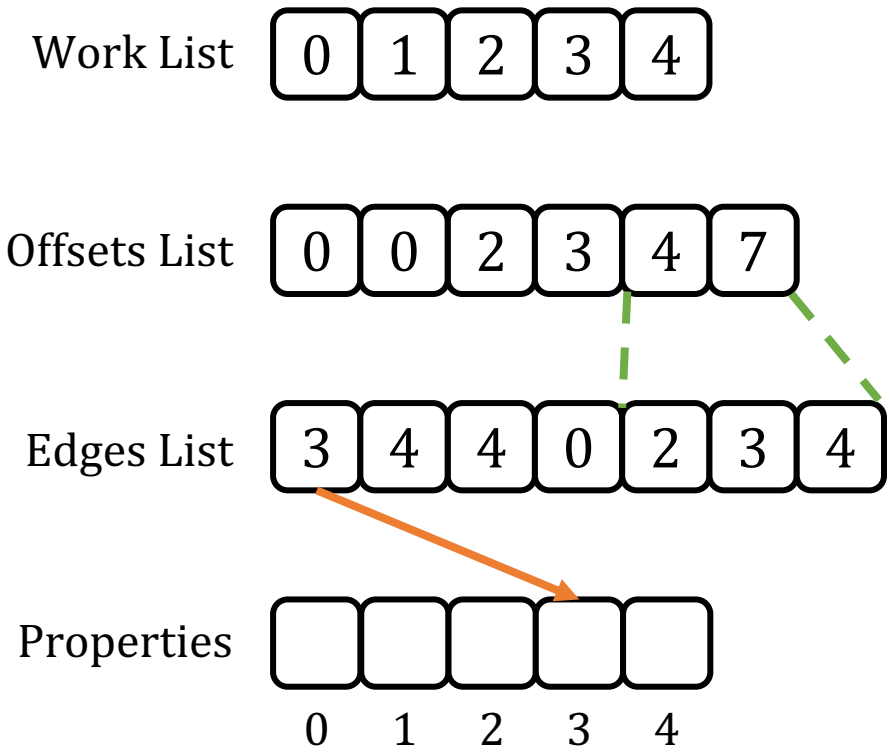
Graph Representation

Compressed sparse row (CSR) format





Traversing the Graph



Two types of indirection:

1. Using a single value
2. Using a range of values

Metadata required for prefetching:

- Type of indirection
- Base address of the next list
- Size of elements in both lists

Graph Prefetching Using MetaSys

Initialize metadata

1. Map metadata IDs to work, offset, and edges lists

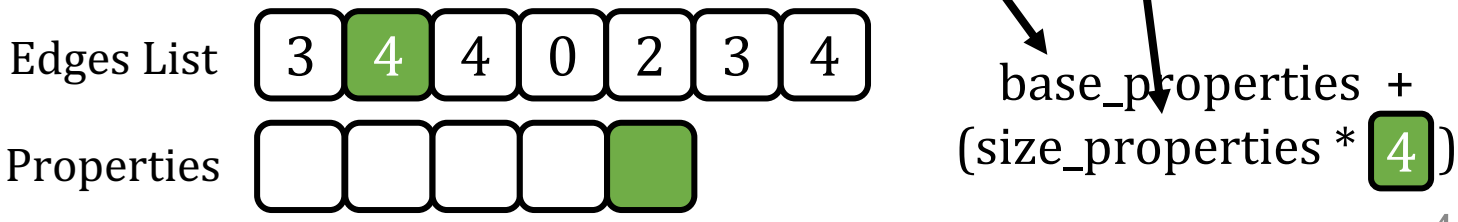
2. **CREATE** metadata to express the characteristics {type, base, base_next, size, size_next} of each list

Prefetch

3. Load/store instruction triggers metadata lookup in prefetcher

find what the application will access next using metadata

Example



Evaluation Methodology

CPU	25 MHz, in-order Rocket core
TLB	16 entry data TLB, LRU policy
L1 D&I$	16 KiB, 4-way, 4 cycle, 64 B cache line, LRU, 2 MSHRs
DRAM	DDR3 1066 MT/s
Metadata Cache	NMRU policy, 128 entries, 38 bits/entry, 512 B granularity
Metadata Table	256 entries, 64 B entry

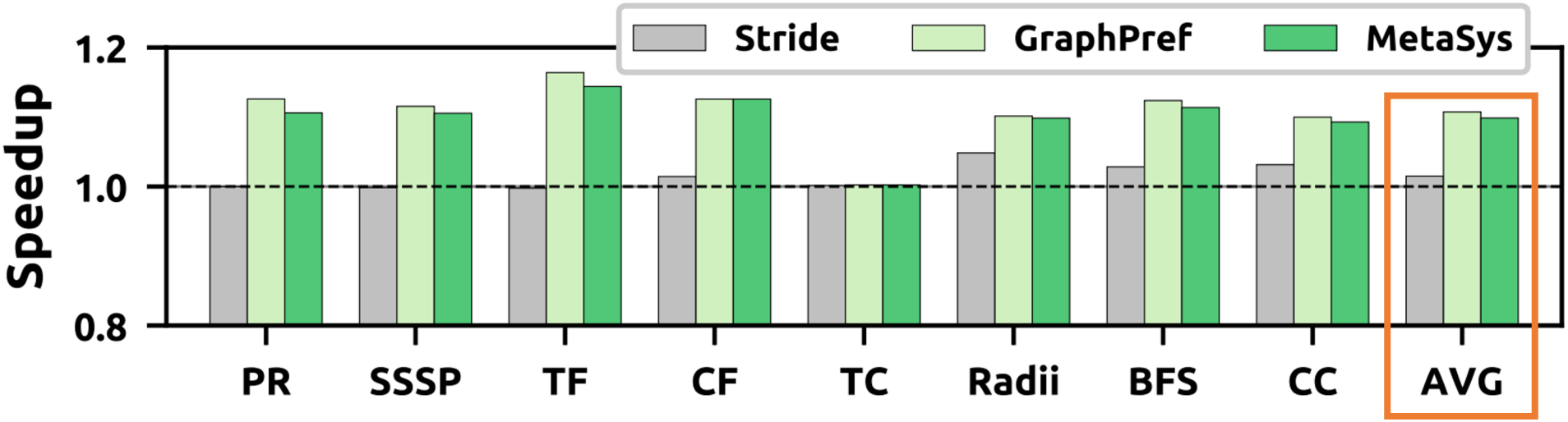
Workloads from the Ligra benchmark (graph applications):

- PageRank (PR), Shortest Path (SSSP), Collaborative Filtering (CF), Teenage Follower (TF), Triangle Counting (TC), Breadth-First Search (BFS), Radius Estimation (Radii), Connected Components (CC)

Evaluated configurations:

- Stride: Baseline system + a hardware stride prefetcher
- GraphPref: Specialized graph prefetcher
- MetaSys: Graph prefetching using MetaSys

Results



MetaSys **improves performance** by **11.2%** on average

Performs **similar to** the specialized prefetcher **GraphPref** (within 0.2%)

Outline

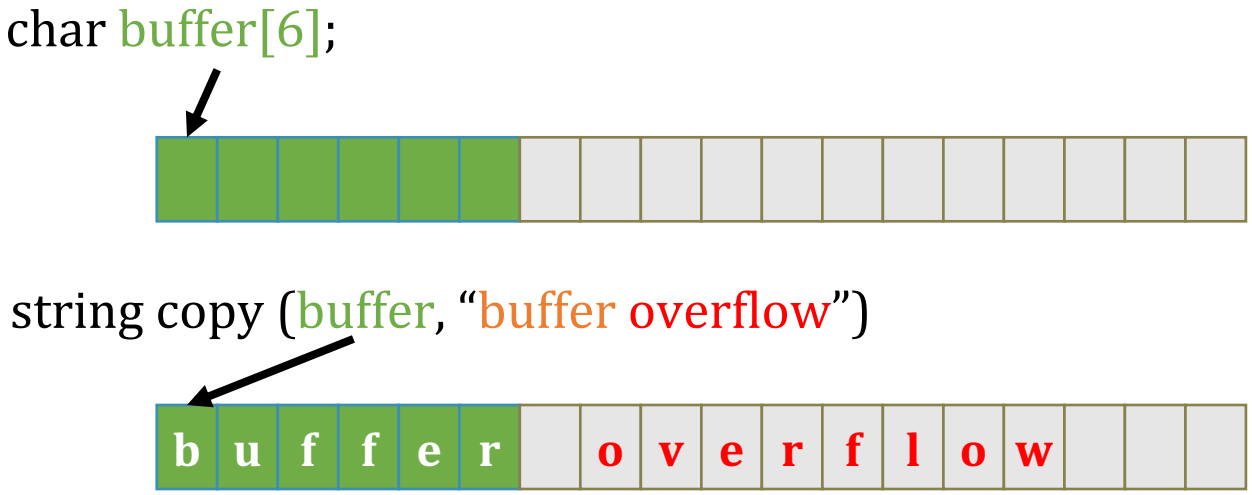
- Background
  - Cross-Layer Techniques
  - Infrastructure for Evaluating Cross-Layer Techniques
- MetaSys
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- **Case Studies**
  - Prefetching
  - **Memory Safety and Protection**
- Characterizing a General Metadata Management System

Memory Safety

Certain programming models are vulnerable to:

- Buffer overflows cause unintentional memory modifications

Memory-unsafe: When there is no bounds checking for pointers



- Software bounds checking: Computationally expensive
- Hardware bounds checking: Too specialized

MetaSys allows bounds checking without additional hardware

Bounds Checking with MetaSys (I)

```
Array A = malloc (16 KB);
MAP(Metadata ID M, Array A);
for (i = 0 ; i < 17 KB ; i++)
    CREATE(M);
    STORE A[i], 1337;
```

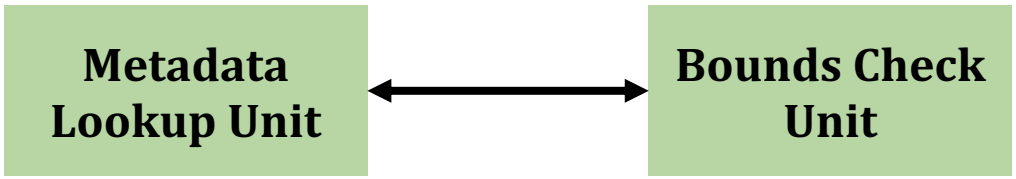
1. Allocate 16 KB of memory

Map Metadata ID M != 0 to Array A

2. Loop exceeds allocated memory range

Execute a CREATE instruction with ID = M before each STORE

3. Eventually, will cause buffer overflow



Bounds Checking with MetaSys (I)

```
Array A = malloc (16 KB);
MAP(Metadata ID M, Array A);
for (i = 0 ; i < 17 KB ; i++)
    CREATE(M);
    STORE A[i], 1337;
```

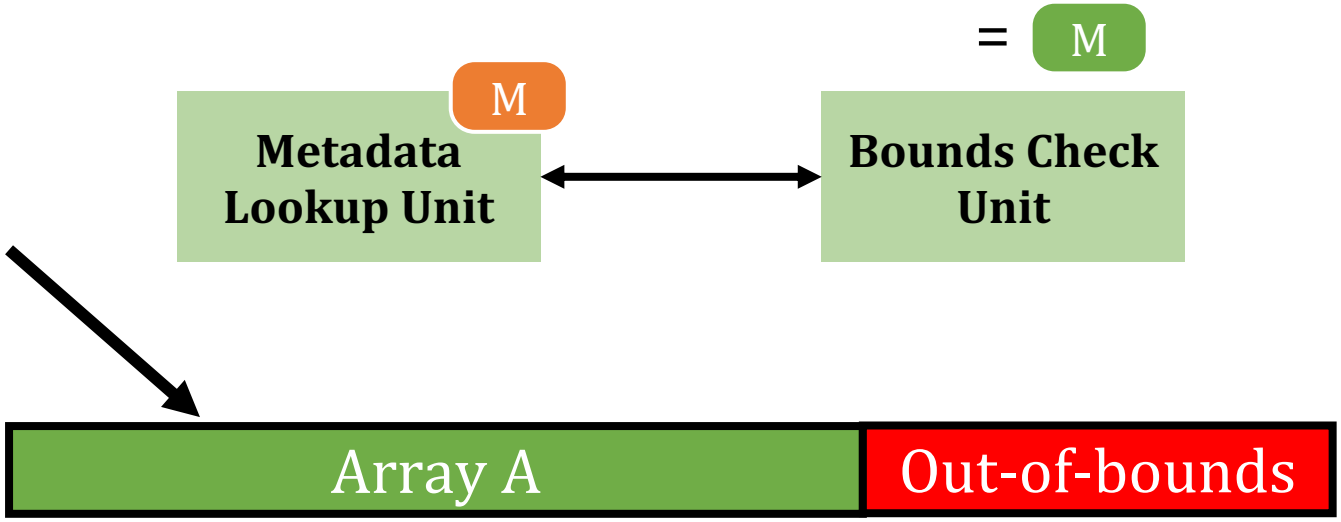
1. Allocate 16 KB of memory

Map Metadata ID M != 0 to Array A

2. Loop exceeds allocated memory range

Execute a CREATE instruction with ID = M before each STORE

3. Eventually, will cause buffer overflow



Bounds Checking with MetaSys (I)

```
Array A = malloc (16 KB);
MAP(Metadata ID M, Array A);
for (i = 0 ; i < 17 KB ; i++)
    CREATE(M);
    STORE A[i], 1337;
```

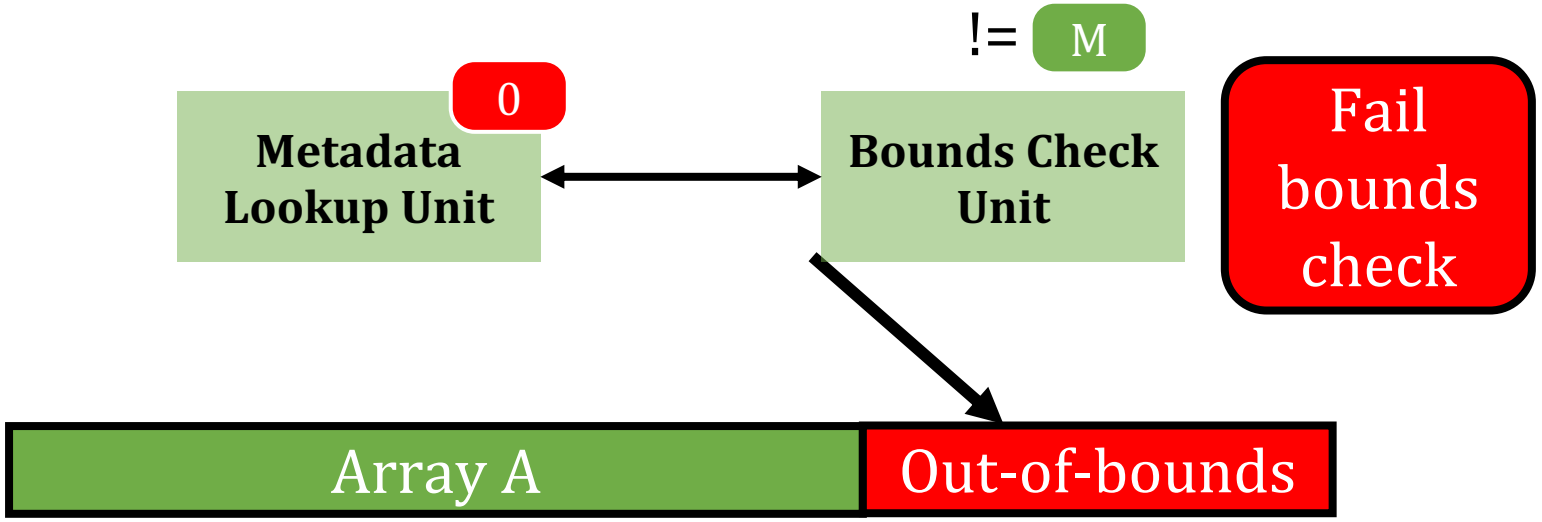
1. Allocate 16 KB of memory

Map Metadata ID M != 0 to Array A

2. Loop exceeds allocated memory range

Execute a CREATE instruction with ID = M before each STORE

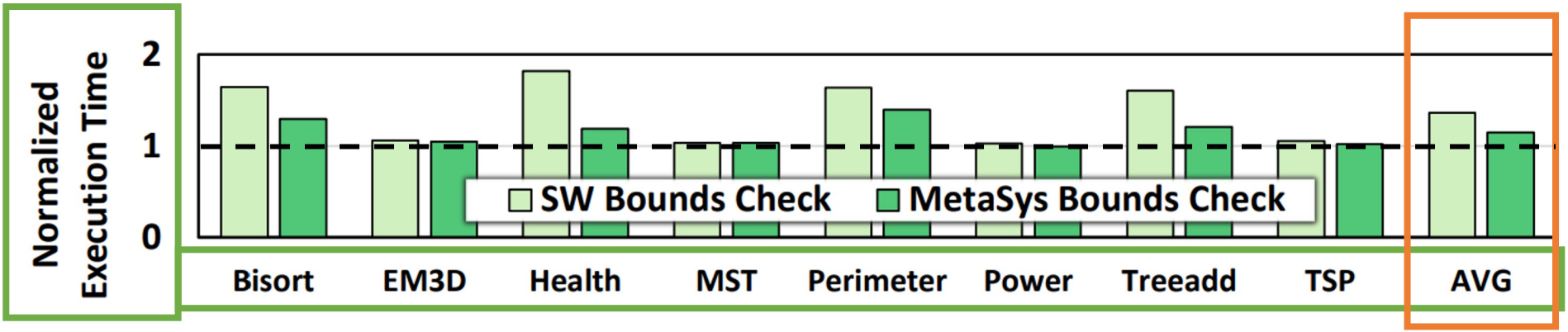
3. Eventually, will cause buffer overflow



Bounds Checking with MetaSys (II)

Evaluation Methodology

- Mapping granularity of 64 B
  - Each 64 B non-contiguous memory location has an ID
- Evaluate Olden benchmarks
- Compare against a software bounds checker



(I) MetaSys bounds checking incurs 14% performance overhead on average
(II) Performs better than the software bounds checker

Return Address Protection with MetaSys

https://arxiv.org/pdf/2105.08123v2

MetaSys: A Practical Open-Source Metadata Management System to Implement and Evaluate Cross-Layer Optimizations

NANDITA VIJAYKUMAR, University of Toronto, Canada
ATABERK OLGUN, ETH Zurich, TOBB ETU, Turkey
KONSTANTINOS KANELLOPOULOS, ETH Zurich, Switzerland
F. NISA BOSTANCI, ETH Zurich, TOBB ETU, Turkey
HASAN HASSAN, ETH Zurich, Switzerland
MEHRSHAD LOTFI, Max Plank Institute, Germany
PHILLIP B. GIBBONS, Carnegie Mellon University, USA
ONUR MUTLU, ETH Zurich, Switzerland

This paper introduces the first open-source FPGA-based infrastructure, MetaSys, with a prototype in a RISC-V system, to enable the rapid implementation and evaluation of a wide range of cross-layer techniques in real hardware. Hardware-software cooperative techniques are powerful approaches to improving the performance, quality of service, and security of general-purpose processors. They are however typically challenging to rapidly implement and evaluate in real hardware as they require full-stack changes to the hardware, system software, and instruction-set architecture (ISA).

Outline

- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- MetaSys
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- **Characterizing a General Metadata Management System**

Characterizing MetaSys

Sources of system overhead:

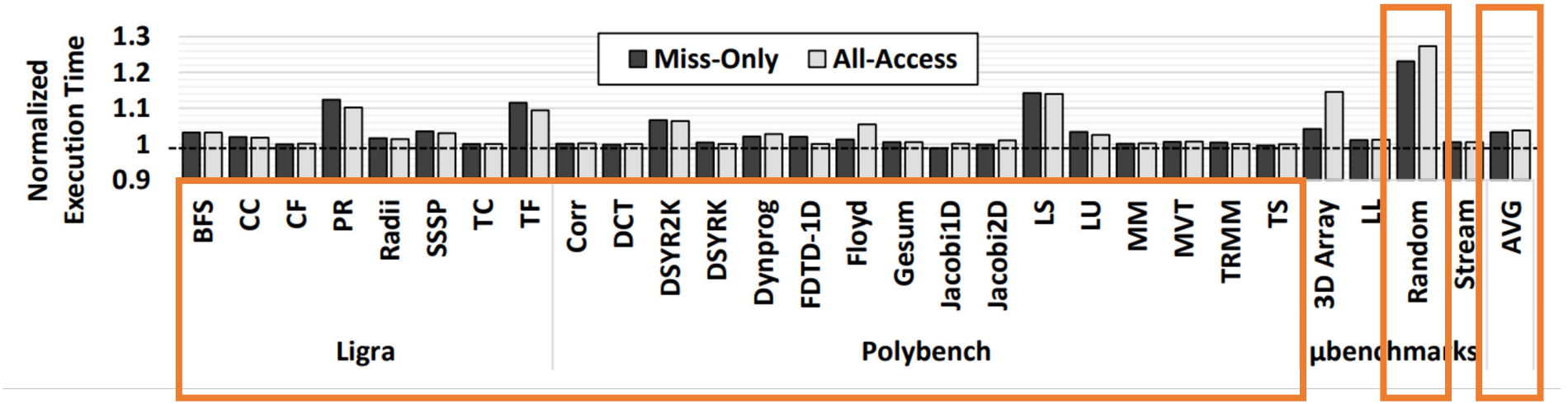
1. Handling dynamic metadata
   - Communicating metadata SW → HW at runtime
2. Metadata management and lookups
   - Components retrieve metadata IDs
3. Scaling to multiple components
   - Multiple components access shared metadata support

Conduct detailed characterization study to understand the overheads

Methodology

- Multiple workloads
  - Polybench, Ligra, and memory-intensive microbenchmarks
- Baseline MetaSys configuration (reminder)
  - 512 B mapping granularity
  - 128 entry metadata mapping cache
  - 256 entry metadata table
  - 64 B metadata
- Stress the metadata management system
  - Perform metadata lookups for every memory access

Overhead of Accessing Metadata

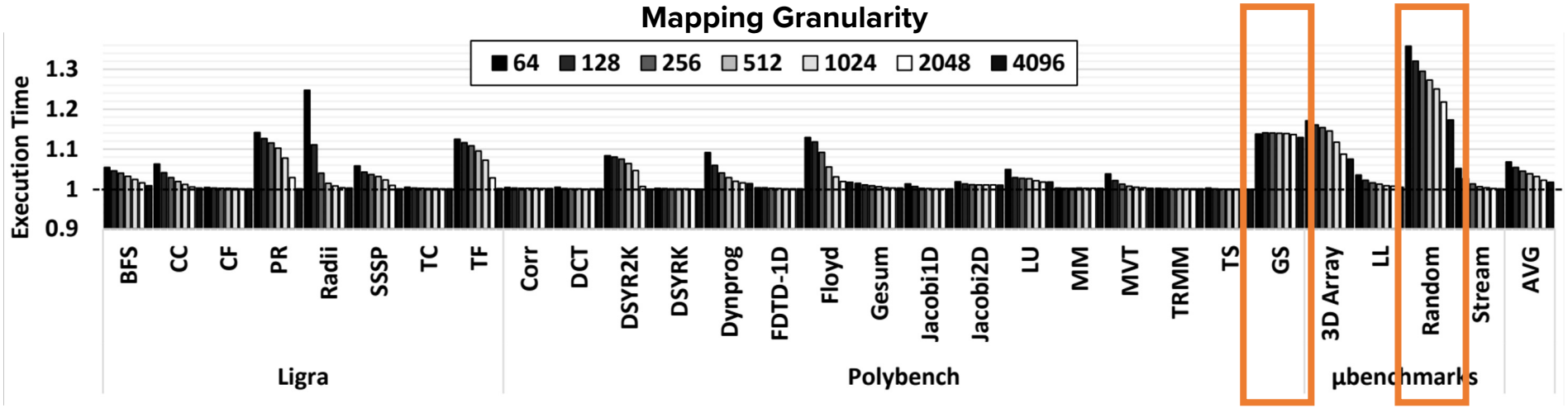


Average **real workload** lookup overhead is low: **2.7%** (~0% min, ~14% max)

Low spatial & temporal locality workloads have high overheads (**Random: 27%**)

Miss-only and all-access are similar in performance (**0.05% difference**)

Effect of Mapping Granularity

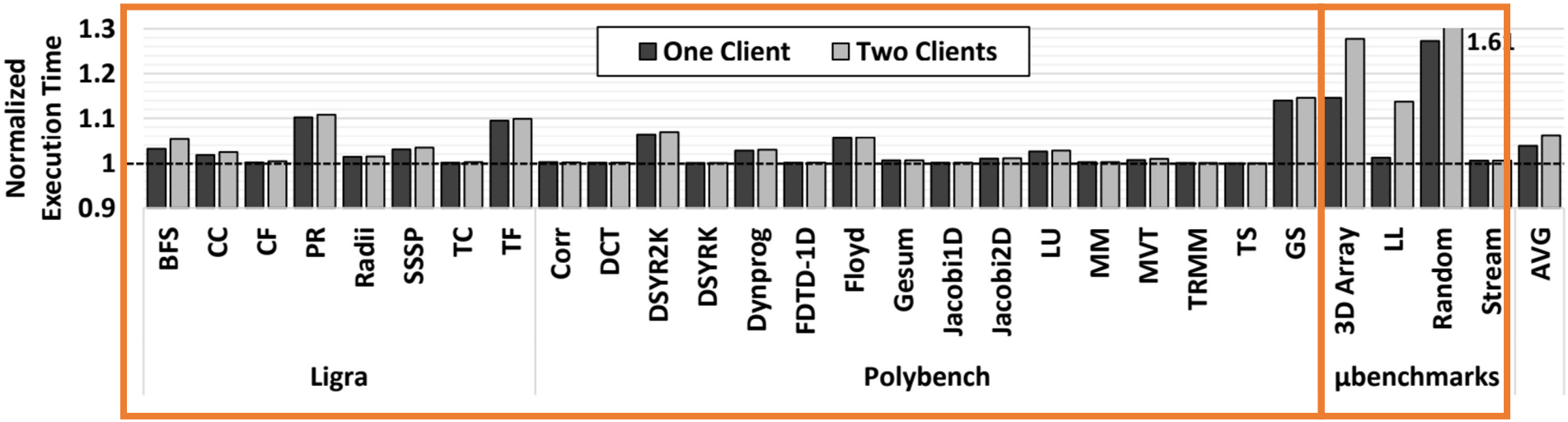


Most workloads have **small performance overhead even** at the **smallest** mapping **granularity**

Some workloads experience **high overhead even** at the **largest** mapping **granularity**

Effect of Multiple Clients

Client 0: Lookup on all memory accesses
Client 1: Lookup on all page table walk requests



Average **real workload** overhead is 0.3%
(over **one client**)

Microbenchmarks experience higher overhead with more clients
In the paper: Investigate mechanisms to alleviate their overheads

Outline

- Background
  - Cross-Layer Techniques
  - Evaluating Cross-Layer Techniques
- MetaSys
  - Overview
  - Components
  - Operation
  - FPGA Prototype
- Case Studies
  - Prefetching
  - Memory Safety and Protection
- Characterizing a General Metadata Management System

More in the Paper

- More details on MetaSys Design
  - OS Support
  - Coherence/Consistency
  - Timing Sensitivity of Metadata Lookups
  - Software Library
  - Area overhead (@ 22nm): 0.03 mm^2
    - 0.02% of an Intel Ivy Bridge CPU core
- In-depth characterization analysis
  - Effects of address translation
  - Effects of Metadata Mapping Cache size
  - Performance overhead of MAP/CREATE instructions

MetaSys Paper

https://dl.acm.org/doi/full/10.1145/3505250

RESEARCH-ARTICLE OPEN ACCESS

MetaSys: A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations

Authors: Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu Authors Info & Claims

ACM Transactions on Architecture and Code Optimization, Volume 19, Issue 2 • June 2022 • Article No.: 26, pp 1–29 • https://doi.org/10.1145/3505250

Published: 24 March 2022 Publication History Check for updates

0 800 View all Formats PDF

Abstract

This article introduces the first open-source FPGA-based infrastructure, MetaSys, with a prototype in a RISC-V system, to enable the rapid implementation and evaluation of a wide range of cross-layer techniques in real hardware. Hardware-software cooperative techniques are powerful approaches to improving the performance, quality of service, and security of general-purpose processors. They are, however, typically challenging to rapidly implement and evaluate in real hardware as they require full-stack changes to the hardware, system software, and instruction-set architecture (ISA).

Executive Summary

Motivation: Hardware-software cooperative (cross-layer) techniques improve performance, quality of service, and security

Problem: Cross-layer techniques are challenging to implement and evaluate in real hardware because they require modifications across the stack

Our Goal is twofold:

1) Enable rapid implementation and evaluation of cross-layer techniques in real hardware

2) Quantify the overheads associated with a general metadata management system

Key Idea:

Develop a metadata management system (**MetaSys**) with hardware and software components that are reusable across cross-layer techniques to minimize programmer effort

Prototype: Xilinx Zedboard prototype using RISC-V Rocket Chip:

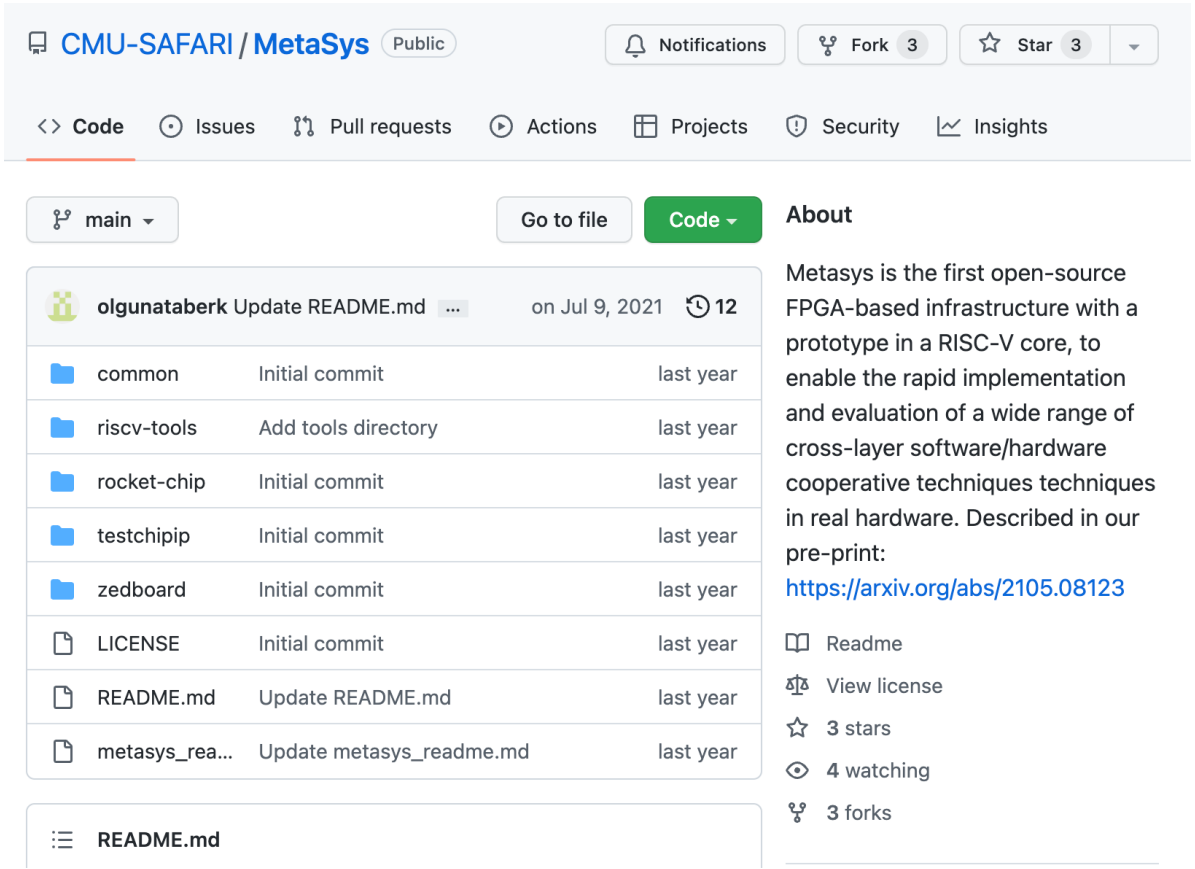
Low on-chip storage (0.2%) and memory storage (0.2%) overhead metadata management

Evaluation: Graph prefetching and memory protection case studies, extensive overhead characterization

- MetaSys-based graph prefetcher performs similar to state-of-the-art specialized prefetcher
- Low-overhead bounds-checking (14%) and return address protection (1.2%)
- Single general metadata management system scales well

MetaSys is Open Source

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



Link to Source Code

Updated Version on ArXiv

https://arxiv.org/abs/2111.00082

arXiv > cs > arXiv:2105.08123

Search... All fields Search

Help | Advanced Search

Computer Science > Hardware Architecture

[Submitted on 17 May 2021 (v1), last revised 2 Jan 2022 (this version, v3)]

MetaSys: A Practical Open-Source Metadata Management System to Implement and Evaluate Cross-Layer Optimizations

Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos, Nisa Bostancı, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

This paper introduces the first open-source FPGA-based infrastructure, MetaSys, with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer techniques in real hardware. Hardware-software cooperative techniques are powerful approaches to improve the performance, quality of service, and security of general-purpose processors. They are however typically challenging to rapidly implement and evaluate in real hardware as they require full-stack changes to the hardware, OS, system software, and instruction-set architecture (ISA).

MetaSys implements a rich hardware-software interface and lightweight metadata support that can be used as a common basis to rapidly implement and evaluate new cross-layer techniques. We demonstrate MetaSys's versatility and ease-of-use by implementing and evaluating three cross-layer techniques for: (i) prefetching for graph analytics; (ii) bounds checking in memory unsafe languages, and (iii) return address protection in stack frames; each technique only requiring ~100 lines of Chisel code over MetaSys.

Using MetaSys, we perform the first detailed experimental study to quantify the performance overheads of using a single metadata management system to enable multiple cross-layer optimizations in CPUs. We identify the key sources of bottlenecks and system inefficiency of a general metadata management system. We design MetaSys to minimize these inefficiencies and provide increased versatility compared to previously-proposed metadata systems. Using three use cases and a detailed characterization, we demonstrate that a common metadata management system can be used to efficiently support diverse cross-layer techniques in CPUs.

Download:

- PDF
- Other formats

(cc) BY

Current browse context:
cs.AR
< prev | next >
new | recent | 2105

Change to browse by:
cs

References & Citations
- NASA ADS
- Google Scholar
- Semantic Scholar

DBLP - CS Bibliography
listing | bibtex
Nandita Vijaykumar
Hasan Hassan
Mehrshad Lotfi
Phillip B. Gibbons
Onur Mutlu

Export Bibtex Citation

Bookmark

MetaSys

A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations

Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos,
F. Nisa Bostancı, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI

ETH Zürich

Carnegie Mellon University

UNIVERSITY OF TORONTO

Backup Slides

OS Support

Support for:

1) managing MMT in physical address space
2) flushing PMTs on context switch
3) updating MMT tags and invalidating MMC on page swaps
4) trap into OS to perform checks (e.g., on bounds check failure)

Coherence/Consistency

- Private metadata tables of private components (e.g., L1 cache) are not coherent
- Private metadata tables of shared components are coherent automatically
  - Updates in PMTs due to CREATEs are visible to all threads immediately
  - Guaranteeing consistency requires use of barriers and fences
- Metadata mapping table shared across threads of the same process
  - Point of coherence is the mapping table, on a MAP, MMC entries in other cores are invalidated
  - Consistency still requires barriers and fences

Timing Sensitivity of Metadata

3 modes on how to handle a lookup:

- **Force stall:** Memory instruction that triggered a lookup cannot commit until the optimization is complete
  - Security use cases
- **No stall:** Metadata lookups never stall the core, but they are always resolved
  - E.g., in page placement, cache replacement
- **Best effort:** Lookups may be dropped before they complete
  - E.g., prefetcher training

Software Library

Table 2. The MetaSys software library function calls.

Library Function Call	**Description**
CREATE(*ClientID*, *TagID*, **meta*)	*ClientID* -> PMT[*TagID*] = **metadata*
MAP(*start**, *end**, *TagID*)	MMT[*start...end*] = *TagID*
UNMAP(*start**, *end**)	MMT[*start...end*] = 0

MetaSys vs XMem

Key difference:

- The whole system is open-sourced

Differences in interfaces:

Table 3. Comparison between MetaSys and XMem interfaces.

Operator	XMem [164]	MetaSys
CREATE	Compiler pragma to communicate static metadata at program load time.	Selects a hardware optimization, dynamically associates metadata with an ID, and communicates both to hardware at runtime (implemented as a new instruction).
(UN)MAP	Associate memory ranges with tag IDs (implemented as new instructions).	Same semantics and implementation as XMem.
(DE)ACTIVATE	Enable/disable optimizations associated with a tag ID (implemented as new instructions).	**Does not exist** as the same functionality can now be done with CREATE.

Existing Cross-Layer Infrastructure

PARD: Enable quality-of-service techniques

- Tag programs with an ID
- Propagate program ID with memory requests
- Hardware enforces QoS requirements on memory requests

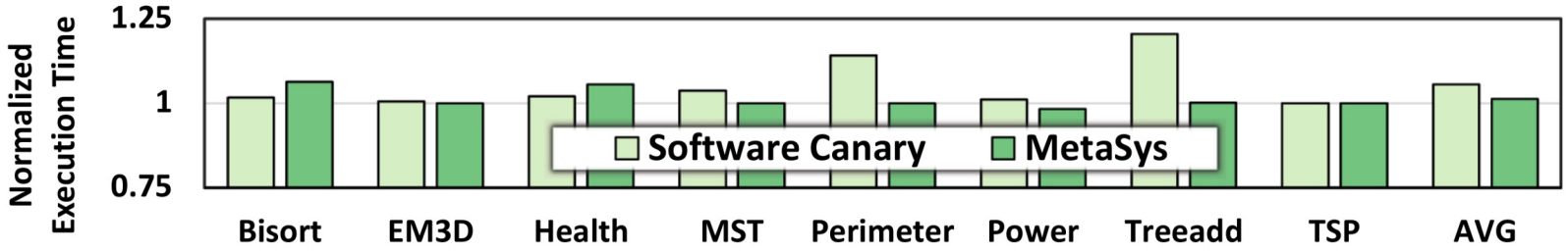
Cheri: Fine-grained memory protection

- Capability registers replace address operands
- Tag capabilities with an ID
- Propagate capability IDs with memory requests
- Hardware enforces bounds checking, pointer integrity, etc.

[Huang+, CARRV'17] [Woodruf+, ISCA'14]

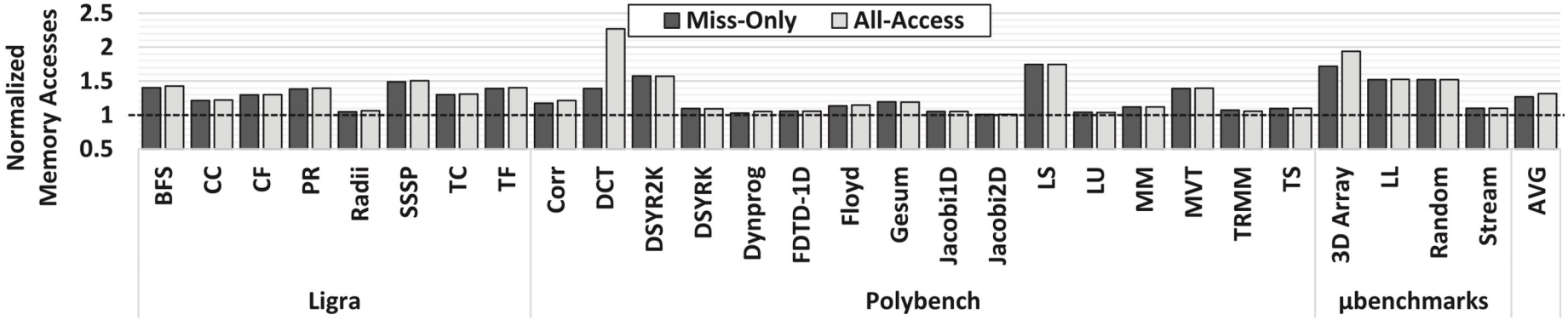
MetaSys: Return Address Protection

1) Map return addresses to metadata id 1
2) Return address protection client performs metadata lookup on every store
3) If a store modifies a non-zero metadata

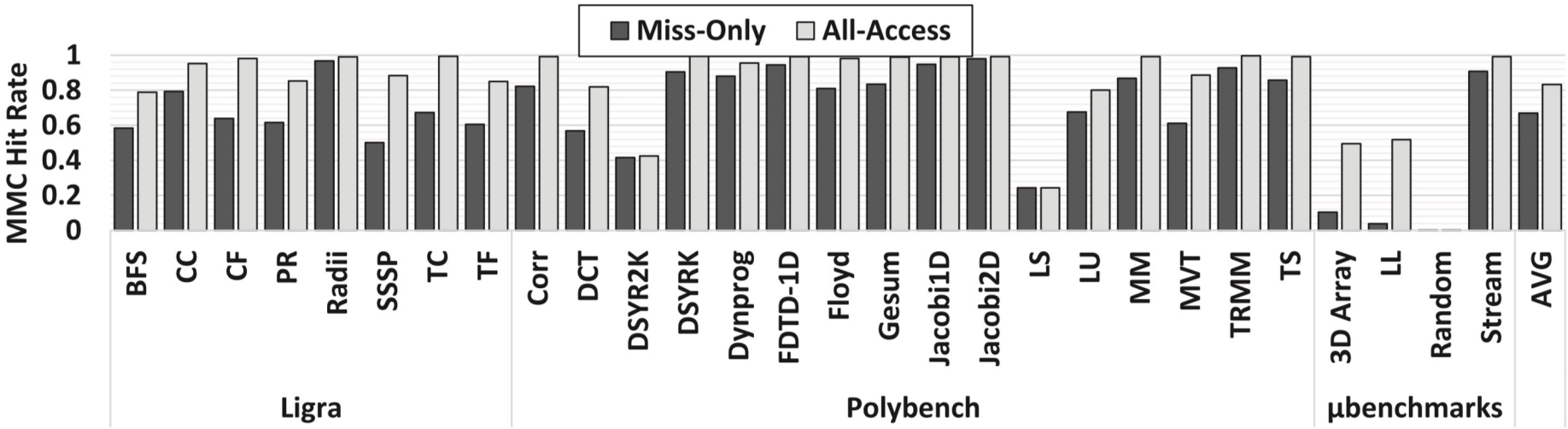


1.2% average performance overhead
(in contrast to software canary's 5.5%)

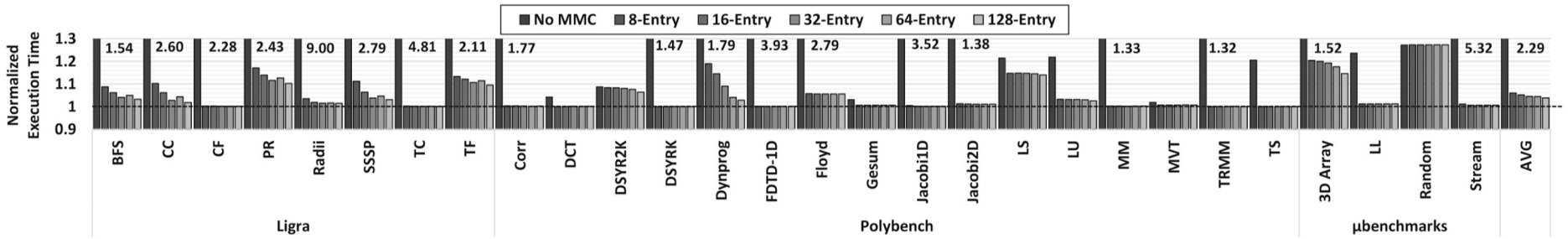
Additional Memory Accesses



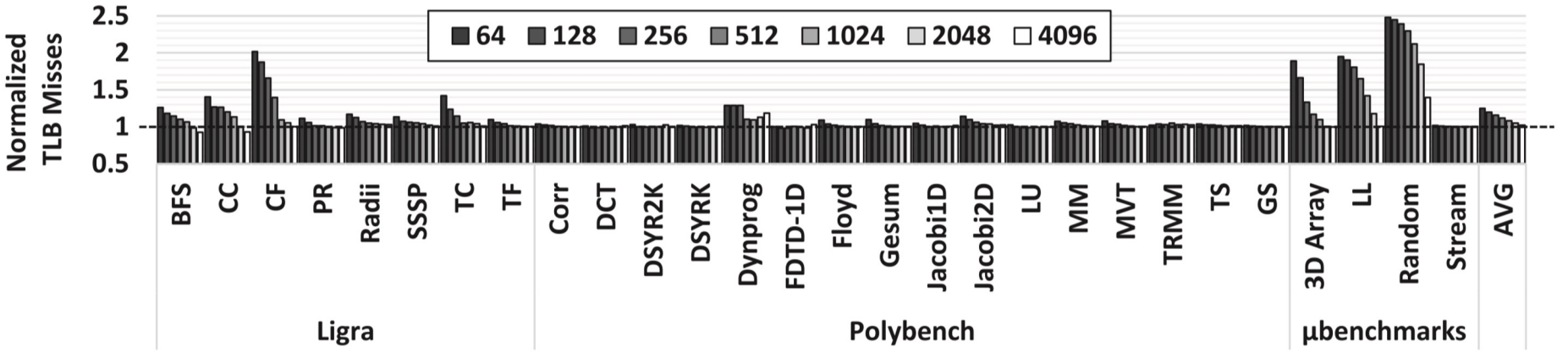
Metadata Mapping Cache Hit Rate



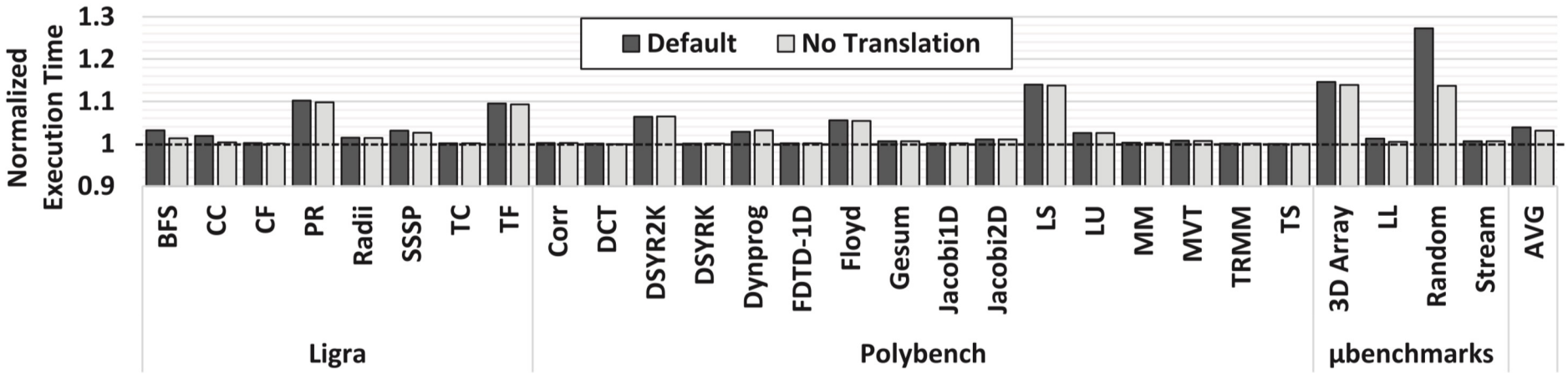
Effect of Metadata Mapping Cache Size



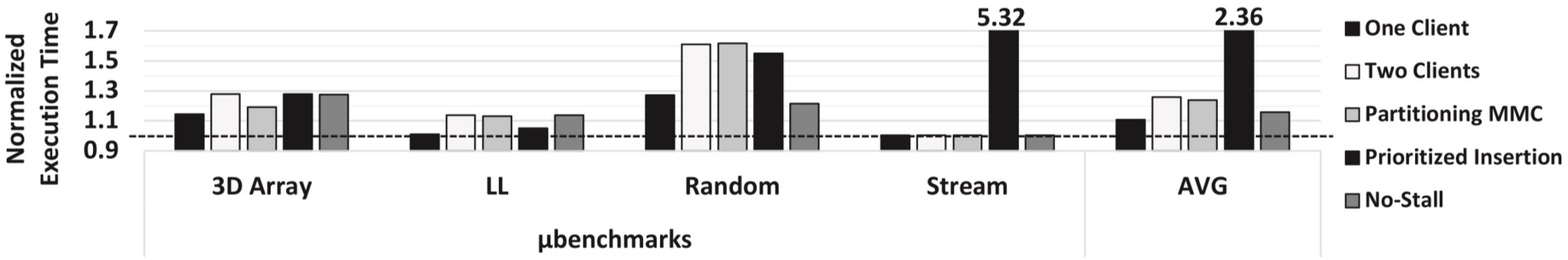
Tagging Granularity vs TLB Misses



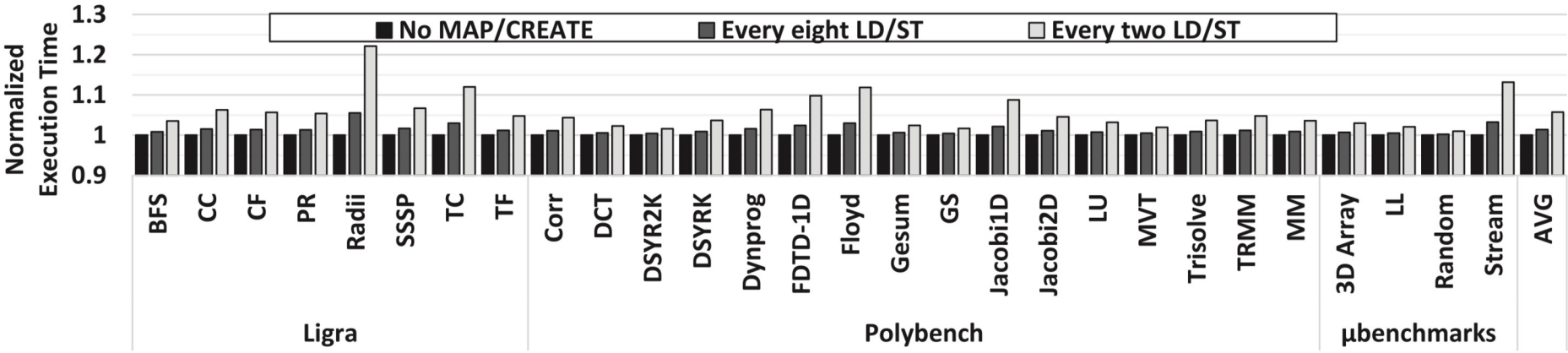
Effect of Address Translation



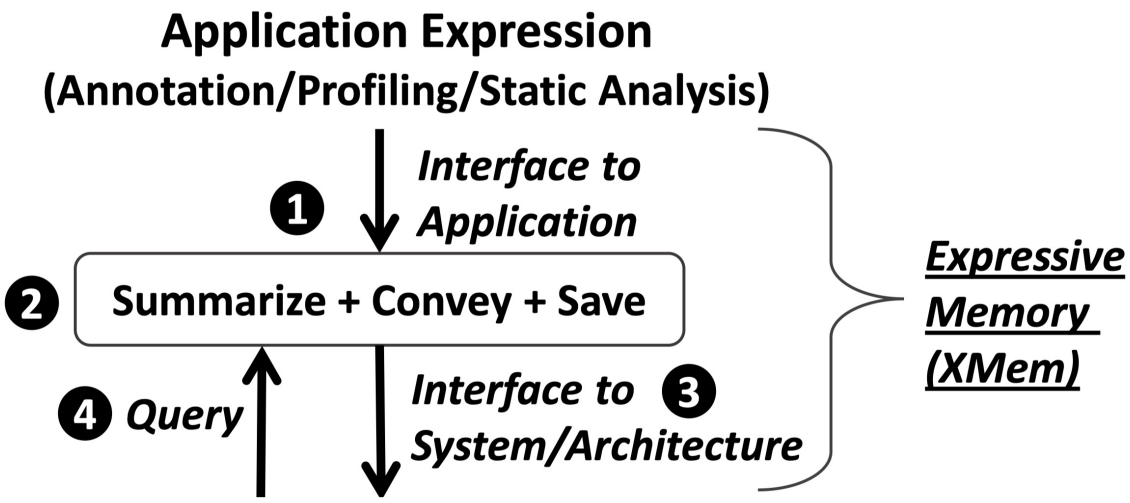
Alleviating Metadata Mapping Cache Contention



MAP/CREATE Instruction Overhead



This work introduces MetaSys:
1) A general metadata management system
2) Various hardware and software components in a **real prototype** to minimize developer effort



[Vijaykumar+, ISCA'18b]