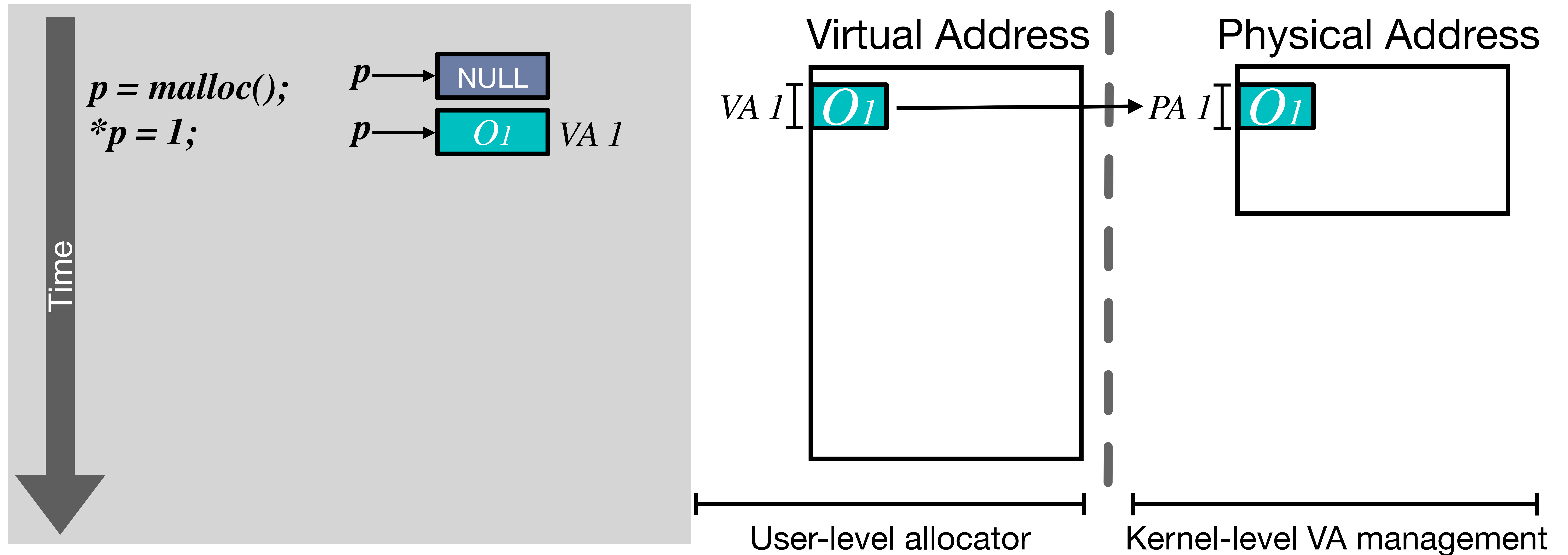
BUDAlloc: Defeating Use-After-Free Bugs by Decoupling Virtual Address Management from Kernel

Junho Ahn, Jaehyeon Lee, Kanghyuk Lee, Wooseok Gwak, Minseong Hwang, Youngjin Kwon

USENIX Security 2024

Use After Free

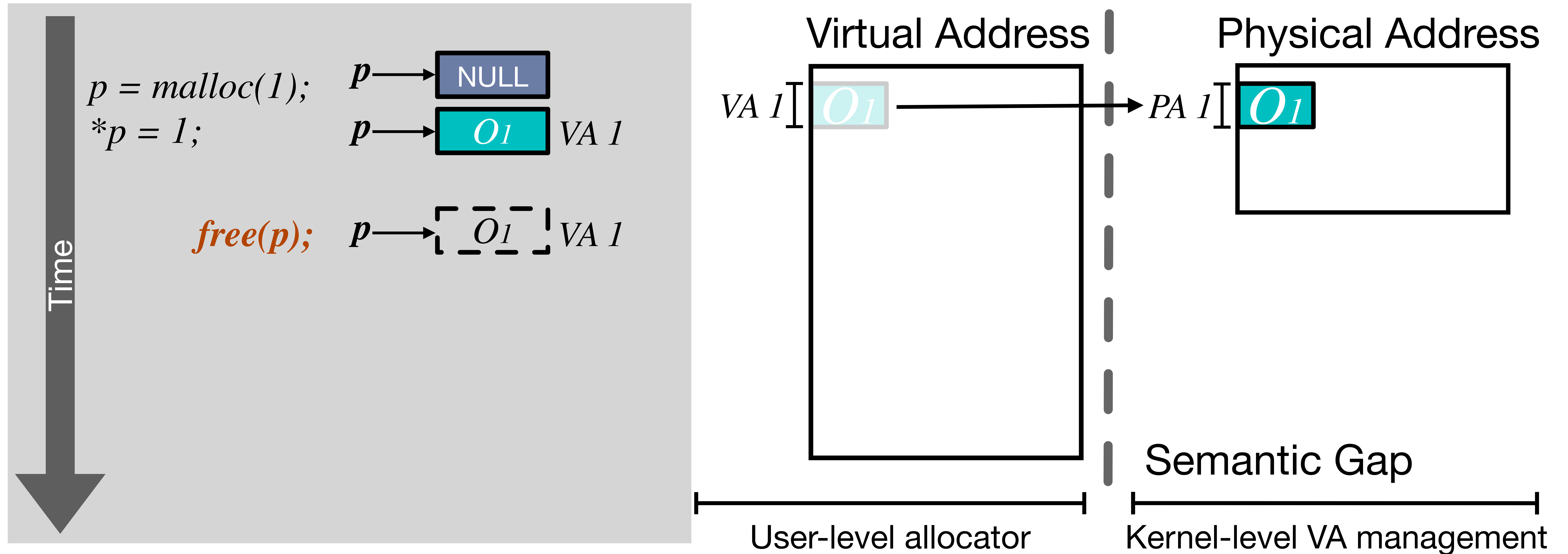


- Memory allocator assigns a virtual address, mapping virtual address to a physical address

KAIST

CASYS
Computer Architecture and Systems Lab

Use After Free

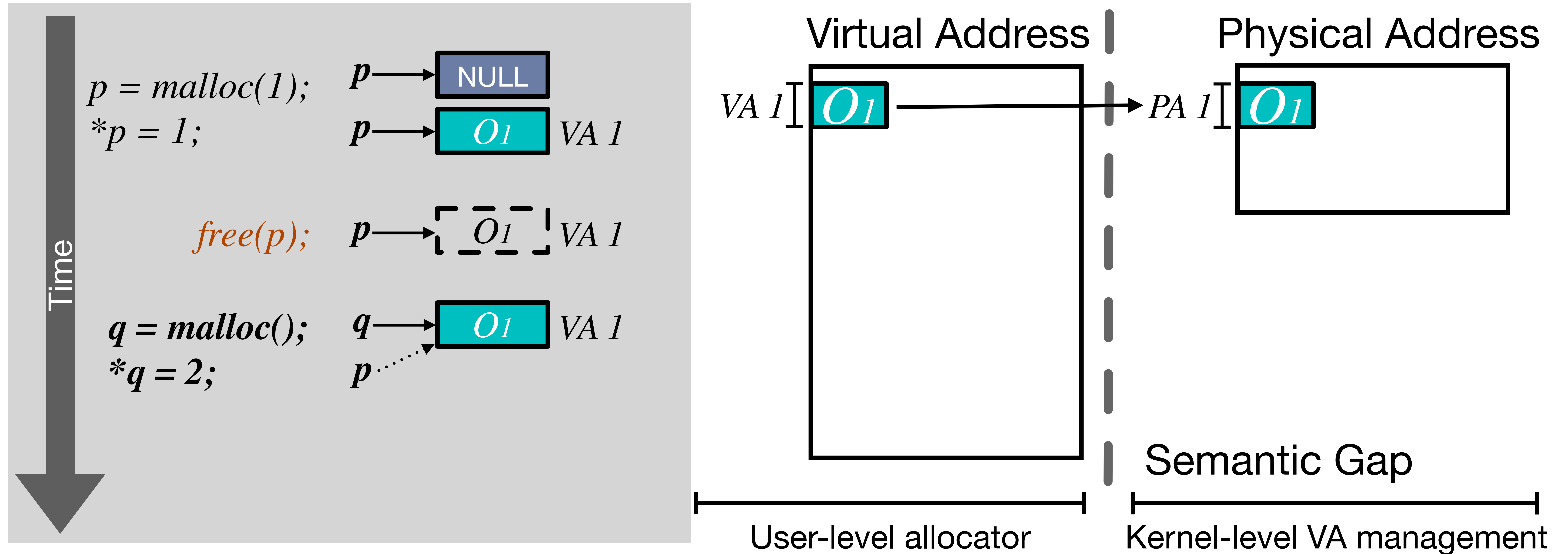


- Memory allocator assigns a virtual address, mapping virtual address to a physical address
- After freeing, the memory allocator retrieves the allocated virtual address

KAIST

CASYS
Computer Architecture and Systems Lab

Use After Free

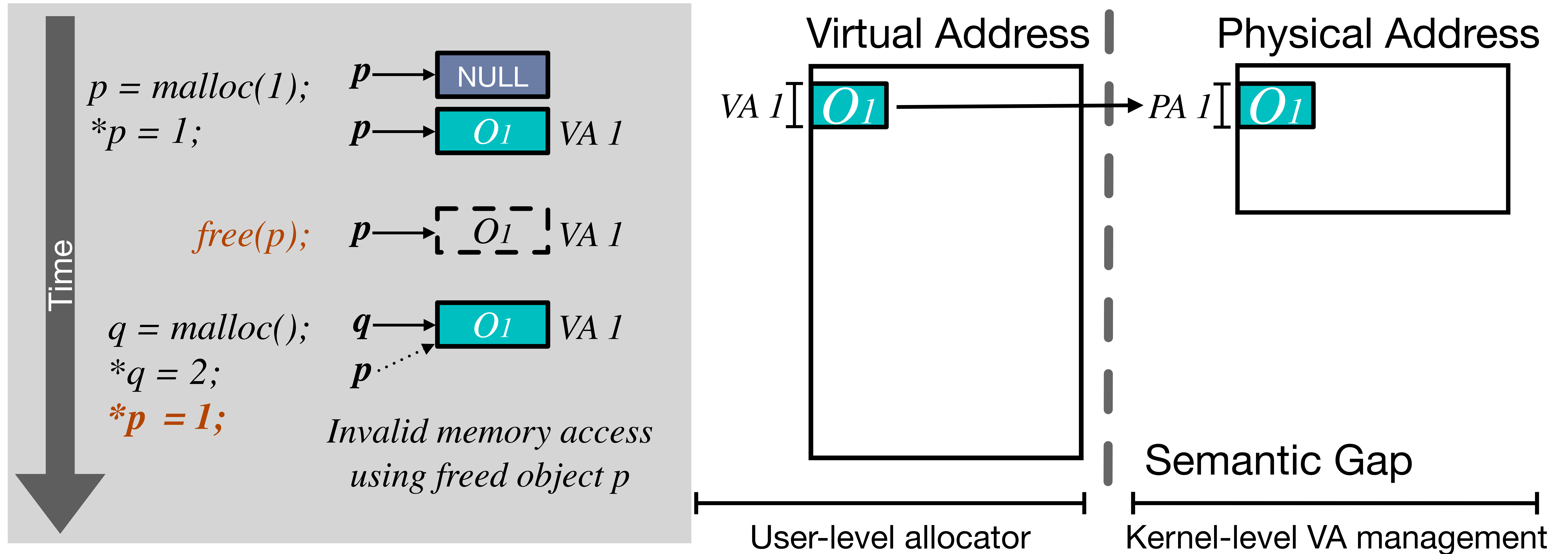


- Use-after-free occurs when program access freed memory
- Attackers can control this freed memory, incurring various malicious behaviors

KAIST

CASYS Computer Architecture and Systems Lab

Use After Free

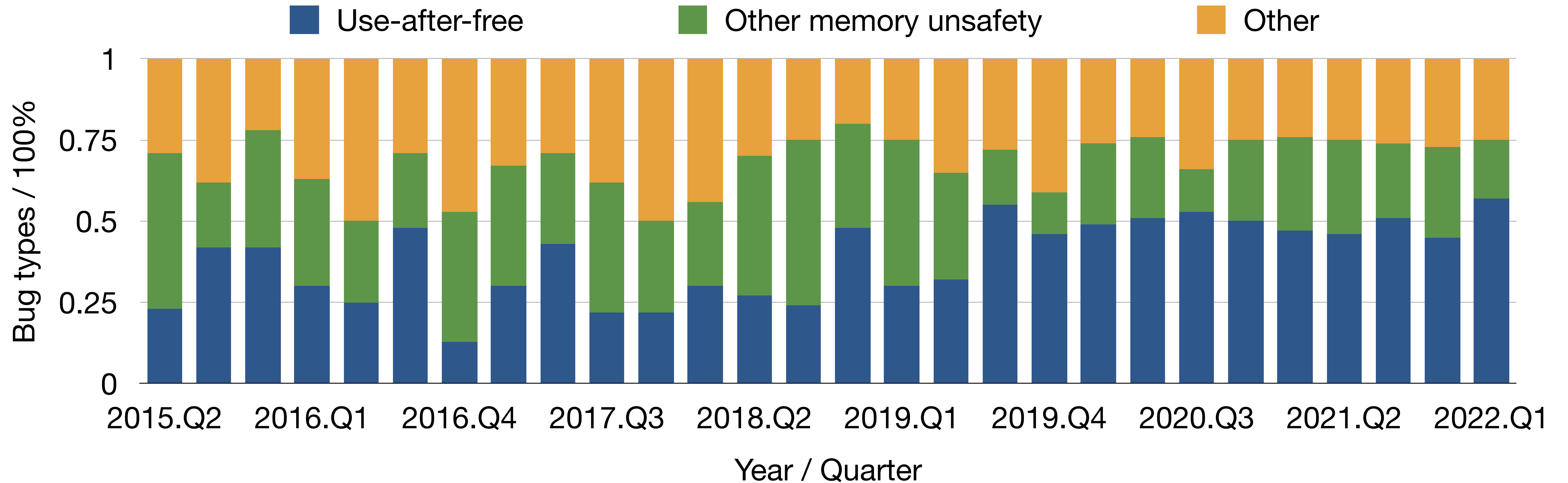


- Use-after-free occurs when program access freed memory
- Attackers can control this freed memory, incurring various malicious behaviors

KAIST

CASYS
Computer Architecture and Systems Lab

Use After Free



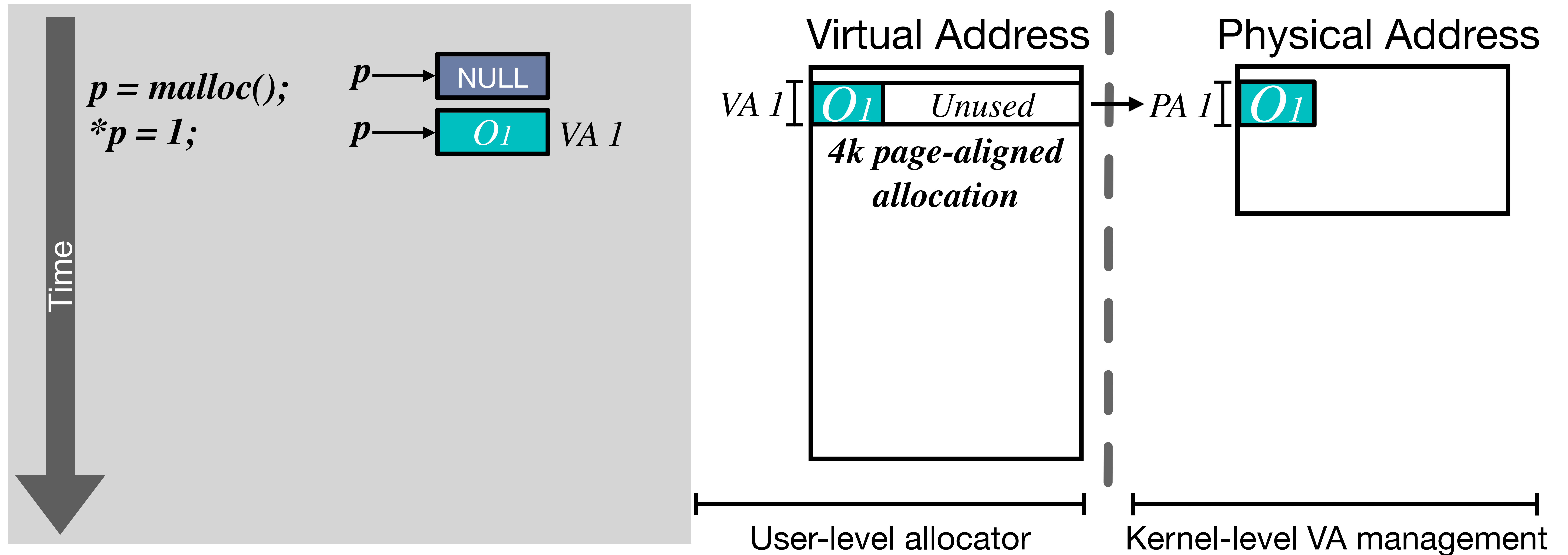
Over 50% of the high-severity bugs discovered in Google Chrome were due to UAF

How to detect and prevent a UAF bug in a C or C++ program?

KAIST

CASYS
Computer Architecture and Systems Lab

One Time Allocator - Detecting UAF Bugs

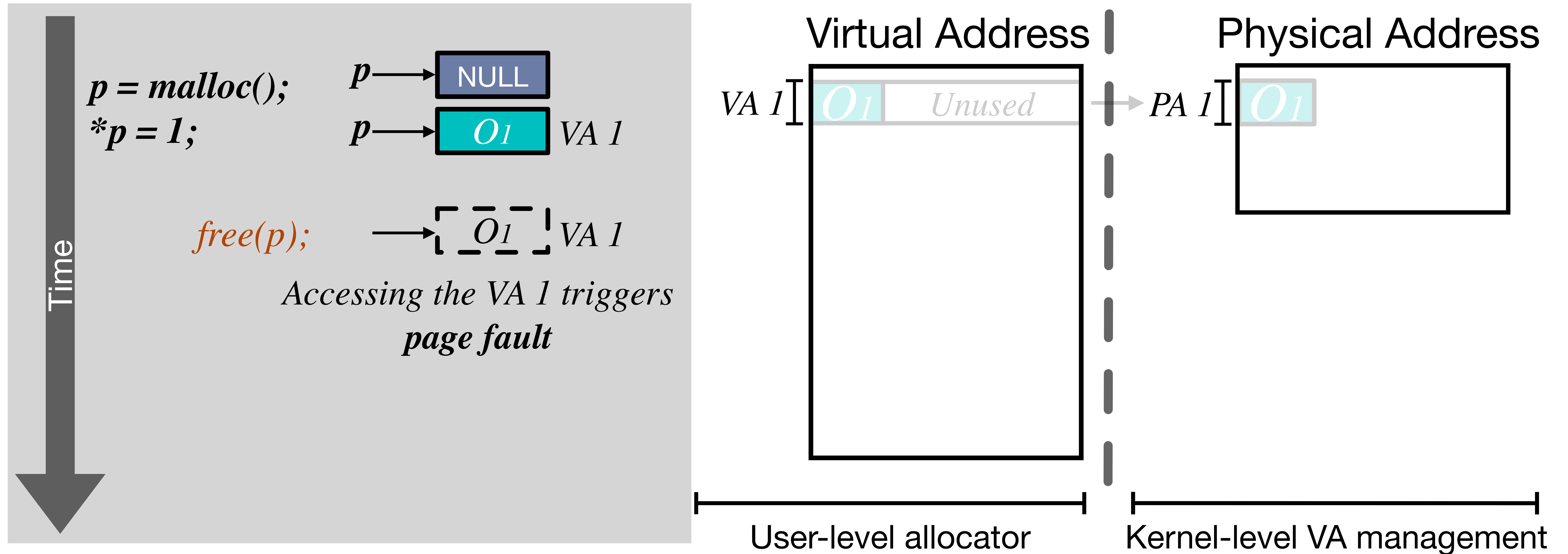


- OTA assigns an object on the single **page-aligned memory**

KAIST

CASYS Computer Architecture and Systems Lab

One Time Allocator - Detecting UAF Bugs

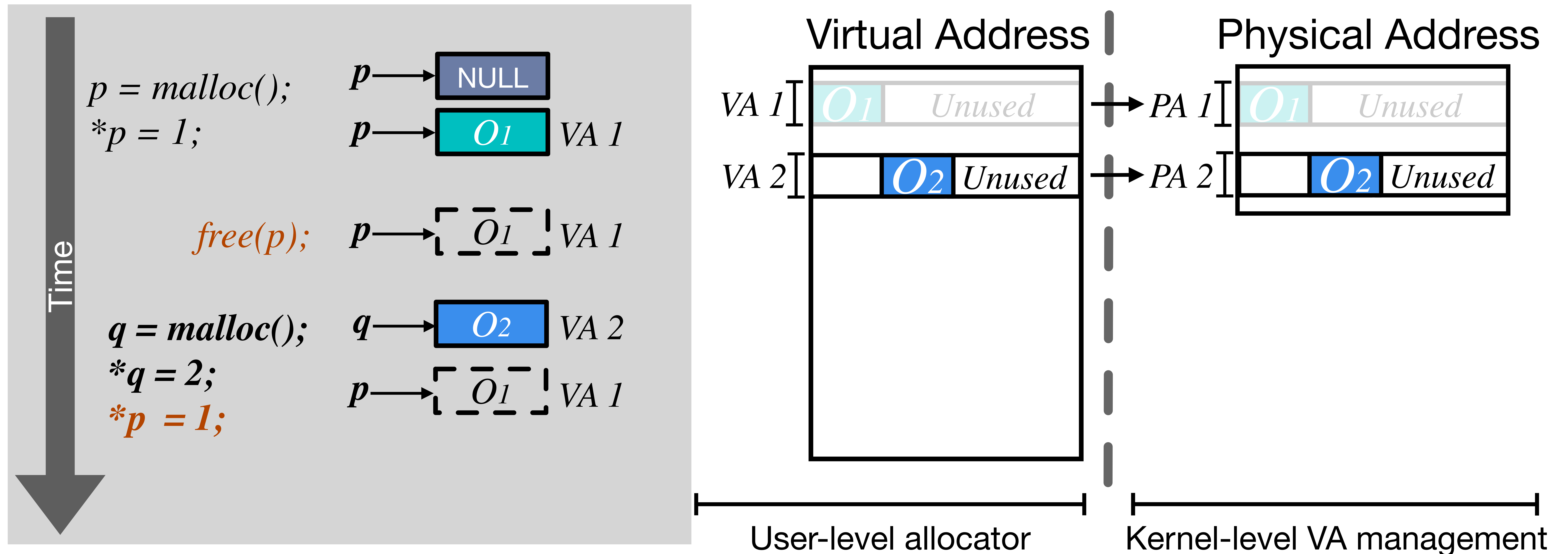


- After free the object, OTA removes the mapping **from the page table**
- Accessing the freed object triggers **page-fault**, which **detects** UAF bug

KAIST

CASYS
Computer Architecture and Systems Lab

One Time Allocator - Detecting UAF Bugs

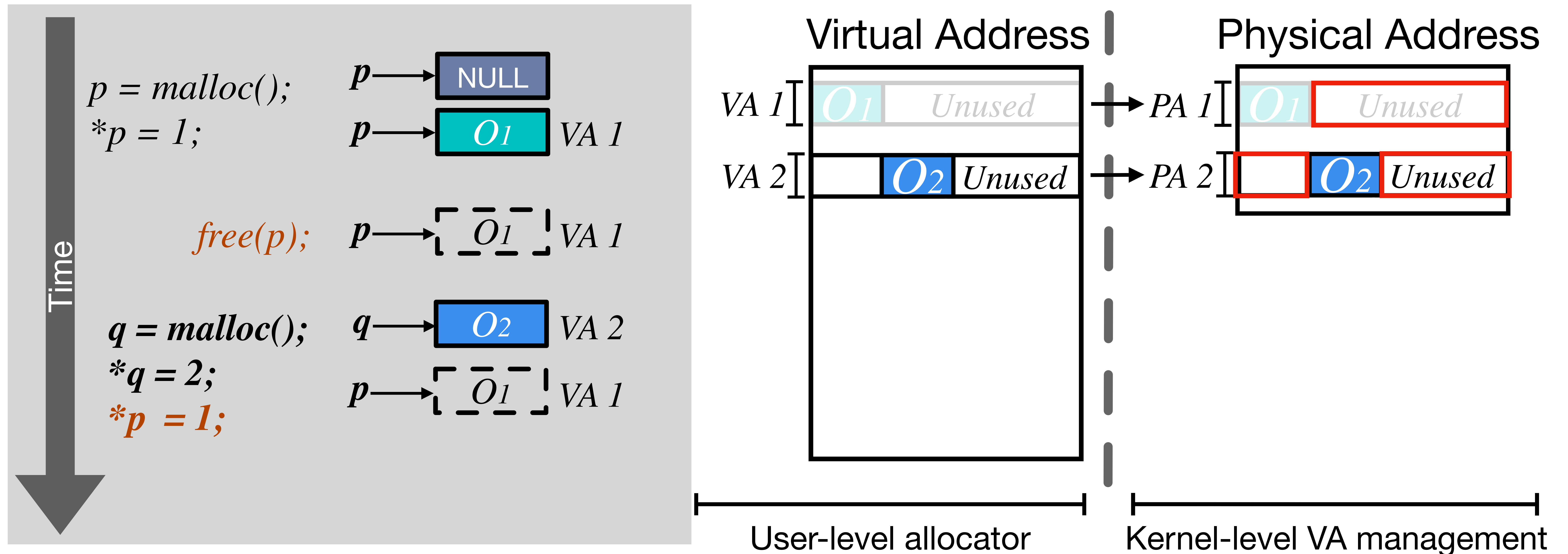


- OTA assigns objects on the virtual address **that has not been previously assigned**
- Ensuring that the deallocated object's address is not reused

KAIST

CASYS
Computer Architecture and Systems Lab

One Time Allocator - Detecting UAF Bugs

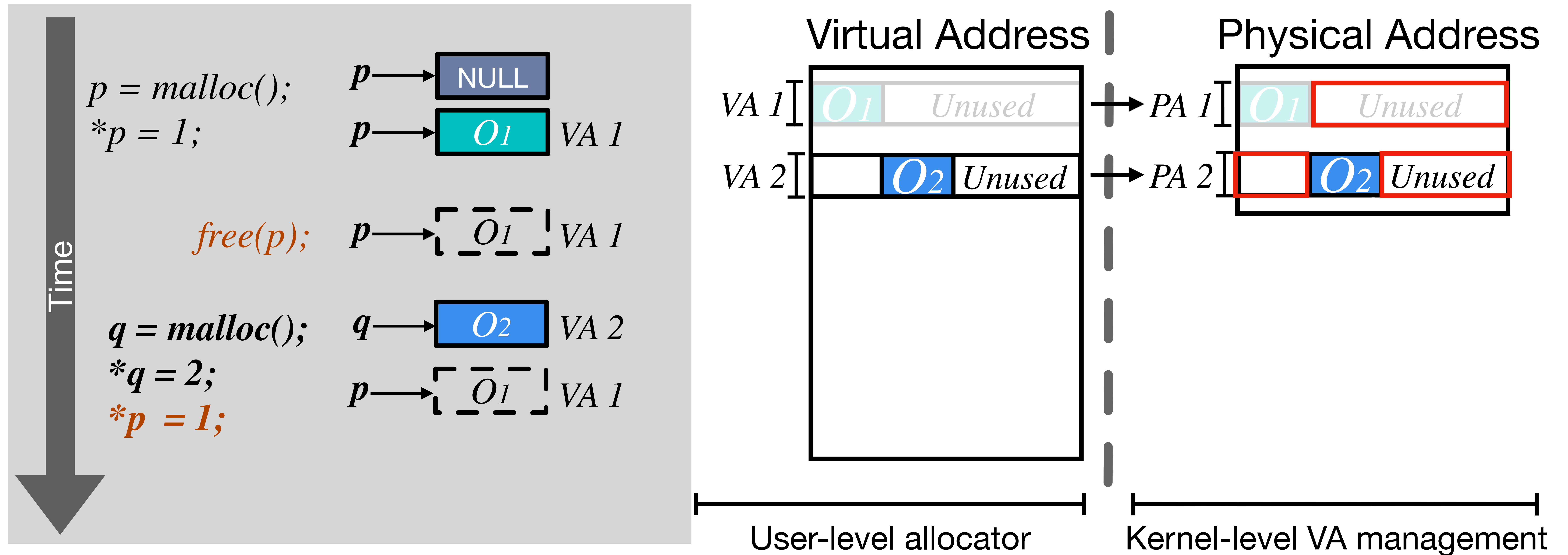


- OTA assigns objects on the virtual address **that has not been previously assigned**
- Ensuring that the deallocated object's address is not reused

KAIST

CASYS
Computer Architecture and Systems Lab

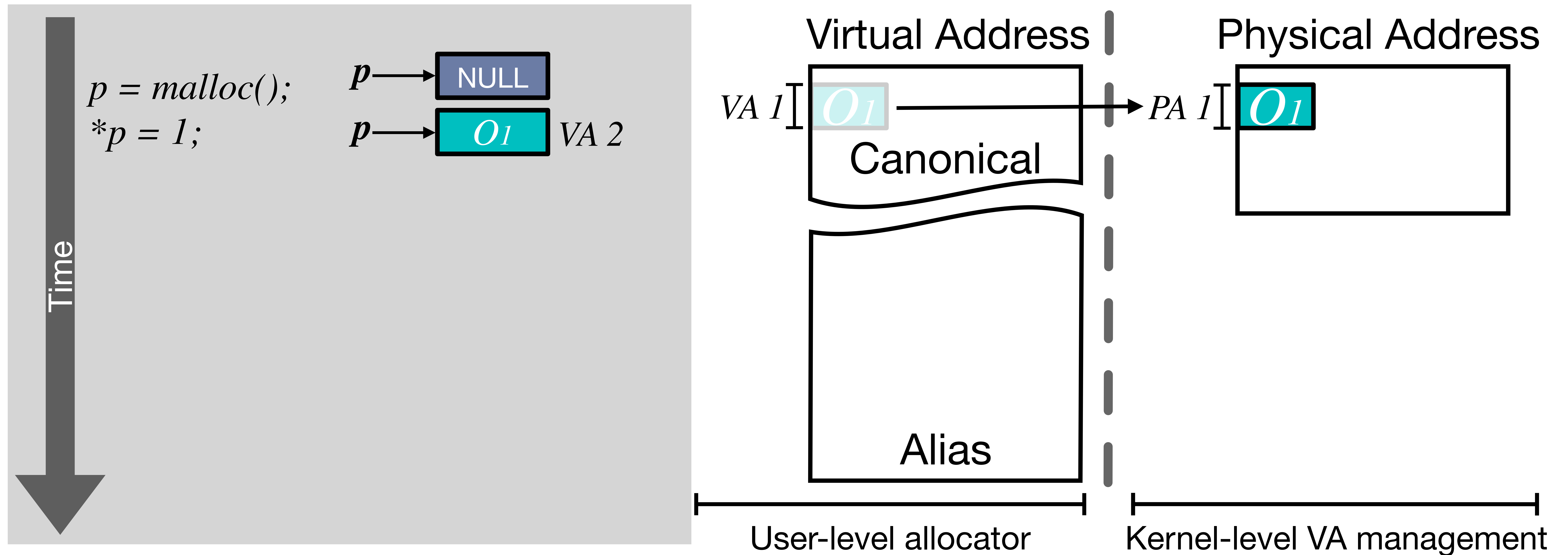
One Time Allocator - Detecting UAF Bugs



- OTA assigns objects on the virtual address **that has not been previously assigned**

Incurs high memory overheads

One Time Allocator - Alias Mapping

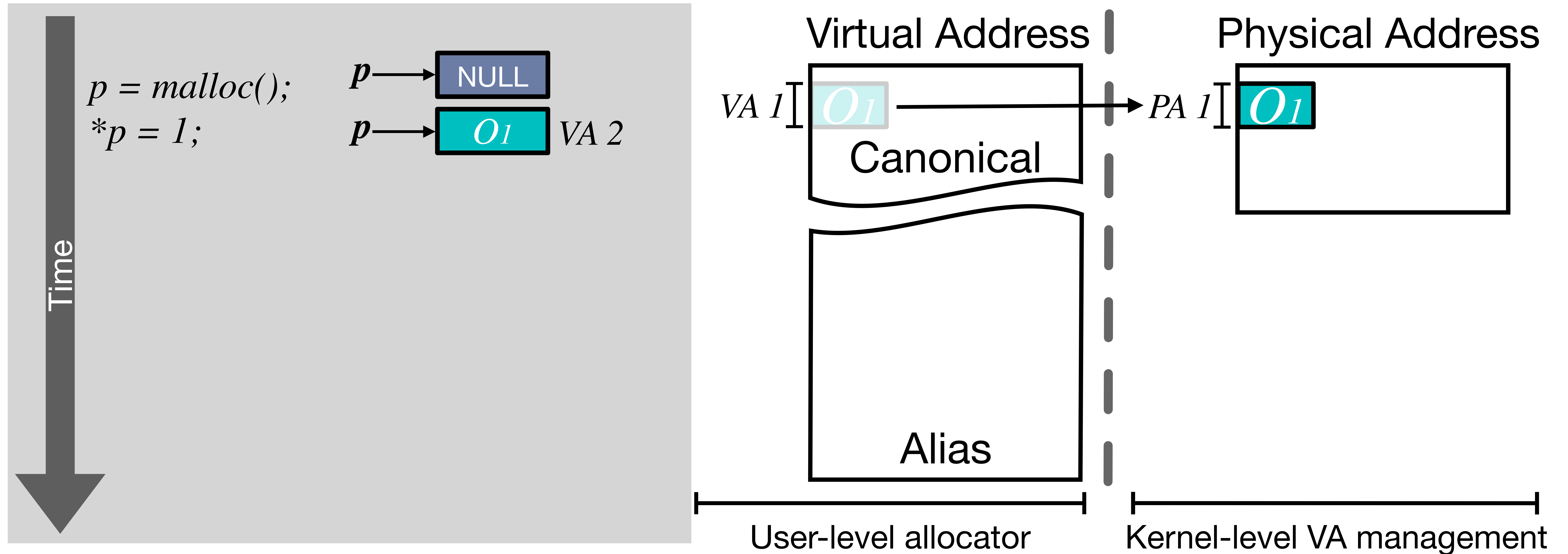


- Alias: **Non-overlapping** VA used by **application**, Canonical: VA used by **allocator**

KAIST

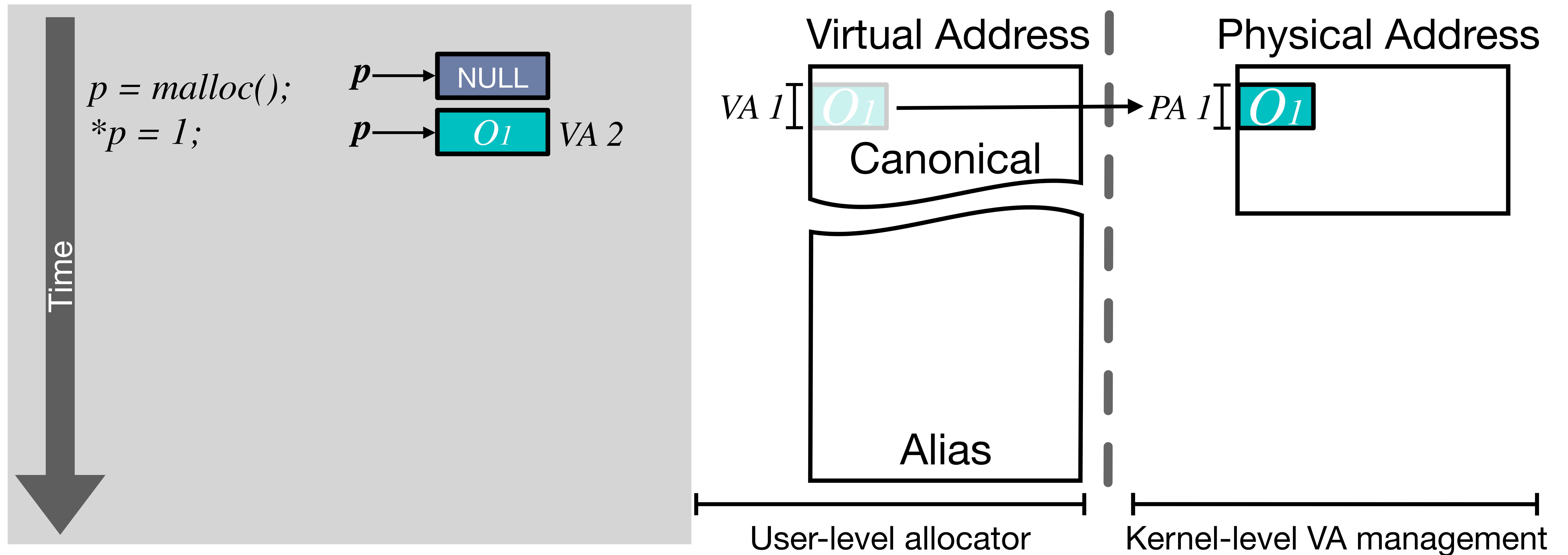
CASYS Computer Architecture and Systems Lab

One Time Allocator - Alias Mapping



- Alias: **Non-overlapping** VA used by **application**, Canonical: VA used by **allocator**

One Time Allocator - Alias Mapping

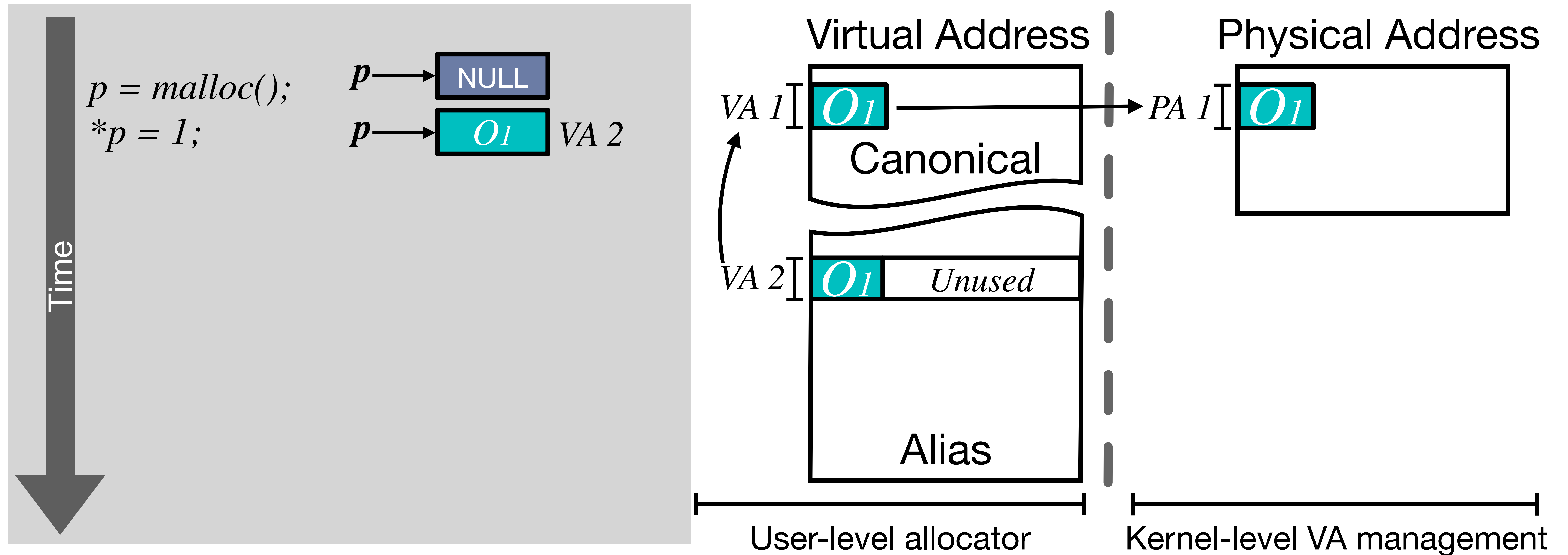


- Alias: **Non-overlapping** VA used by **application**, Canonical: VA used by **allocator**

KAIST

CASYS Computer Architecture and Systems Lab

One Time Allocator - Alias Mapping

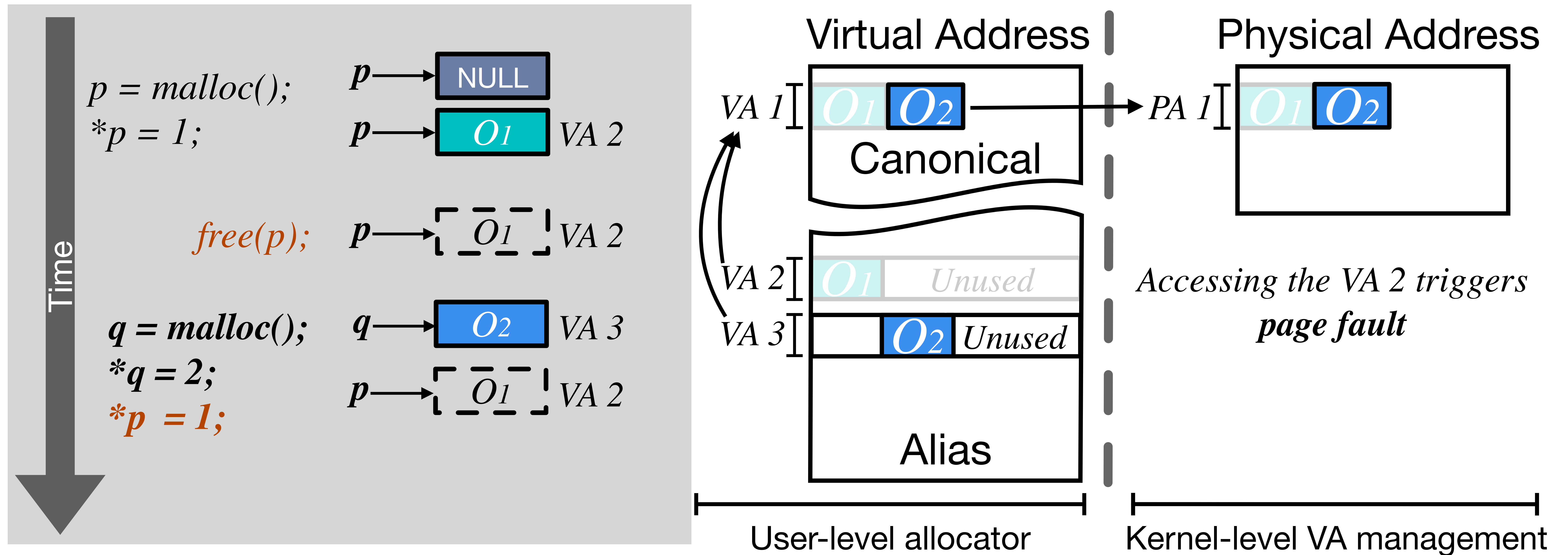


- Alias: **Non-overlapping** VA used by **application**, Canonical: VA used by **allocator**
- OTA maps **multiple page-aligned alias address to canonical address**

KAIST

CASYS Computer Architecture and Systems Lab

One Time Allocator - Alias Mapping

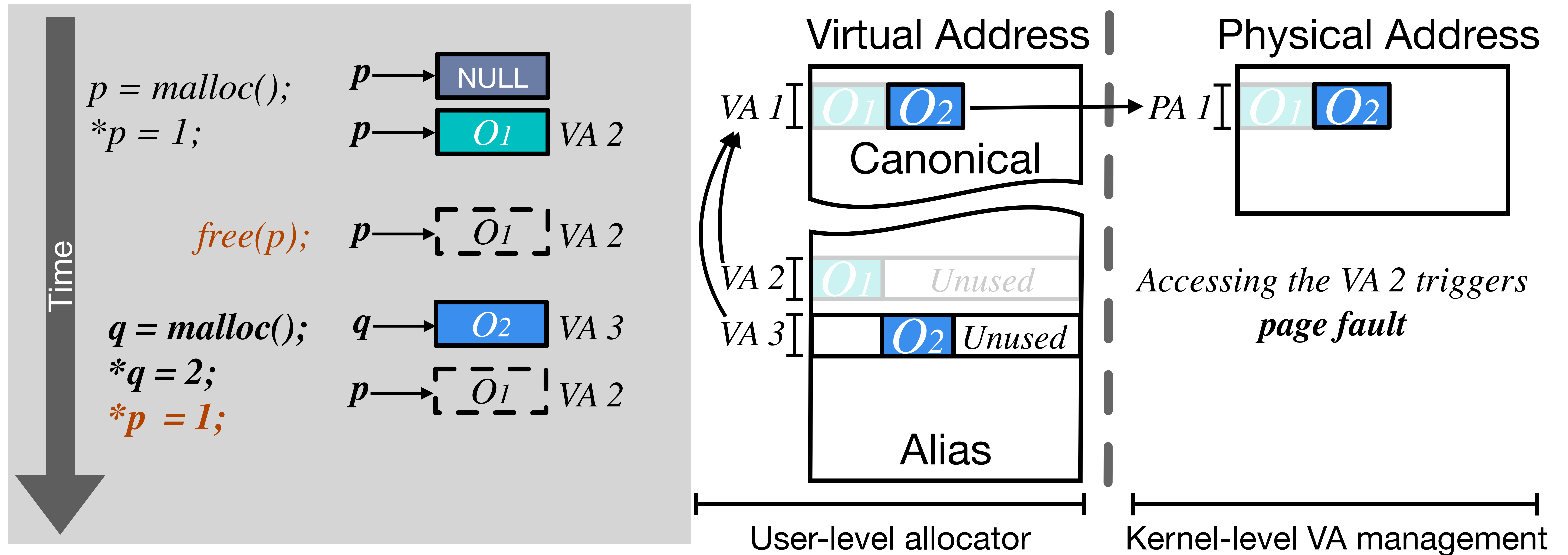


- Alias mapping reuses the freed canonical address, thus **reduce the memory overheads**
- Despite the reuse, alias mapping still can **detect** UAF bugs

KAIST

CASYS
Computer Architecture and Systems Lab

One Time Allocator - Alias Mapping



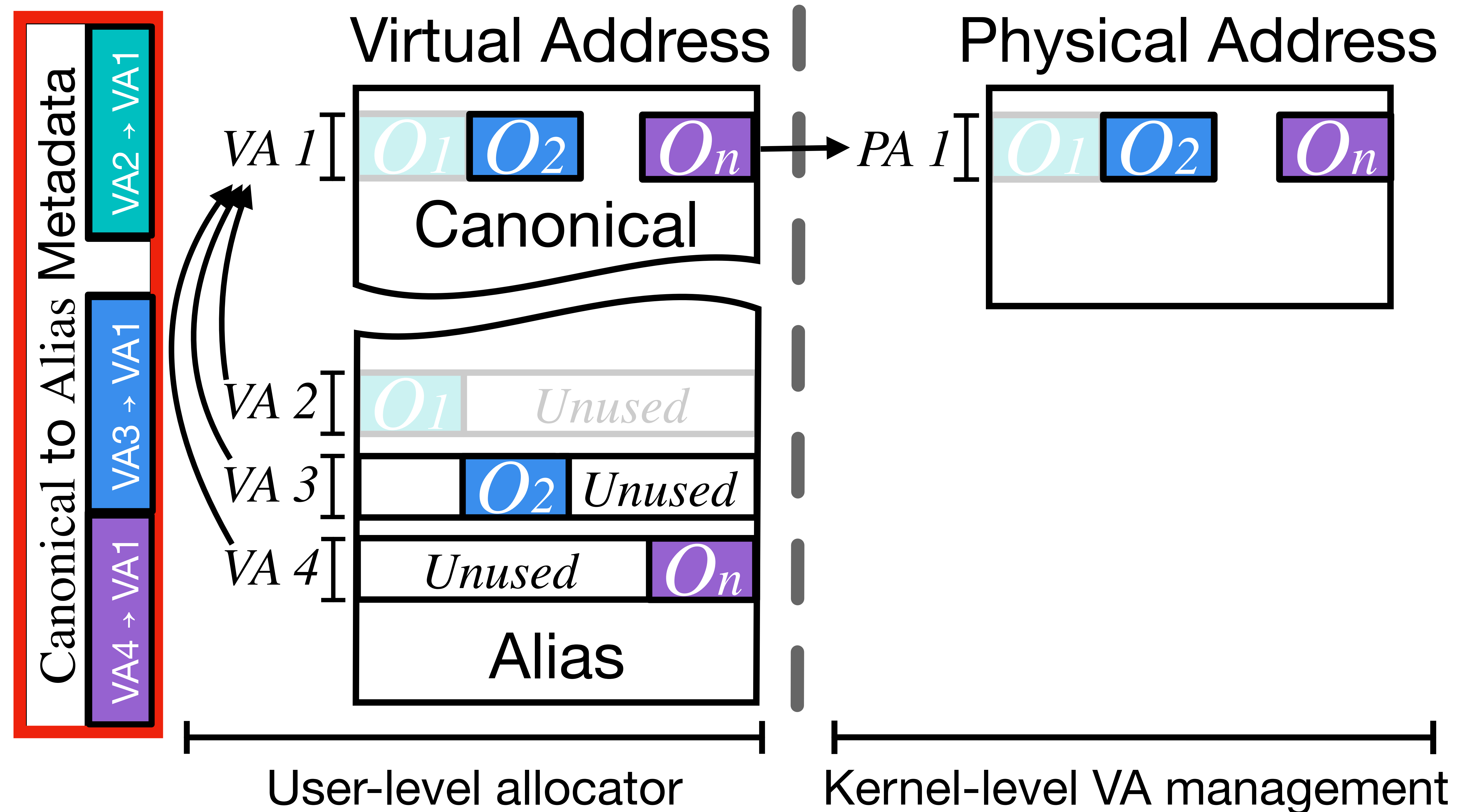
- Alias mapping reuses the freed canonical address, thus **reduce the memory overheads**

Incurs high performance overheads

KAIST

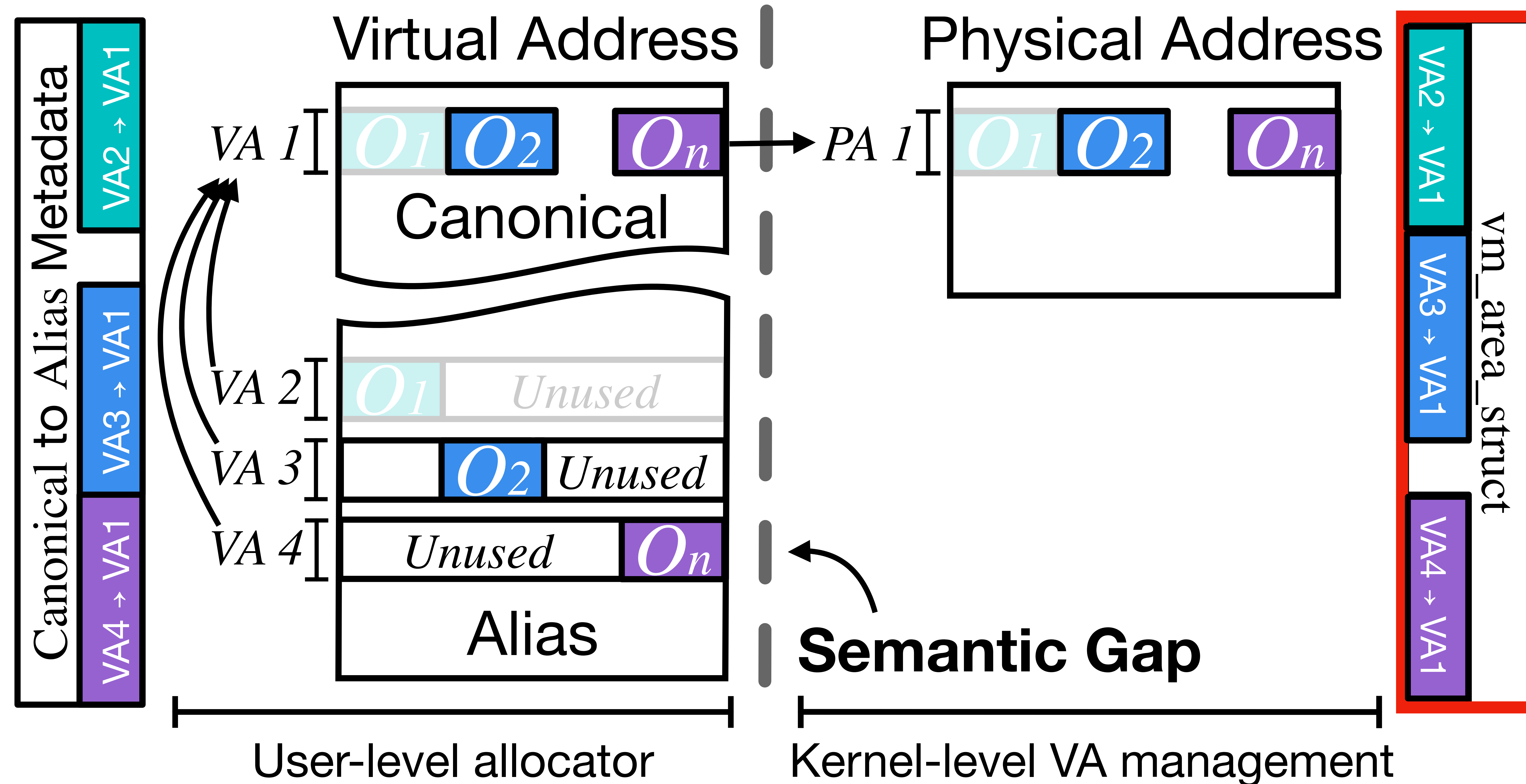
CASYS Computer Architecture and Systems Lab

Performance Overhead due to Semantic Gap



- OTA manages the alias to canonical mapping using the **user-level metadata**

Performance Overhead due to Semantic Gap

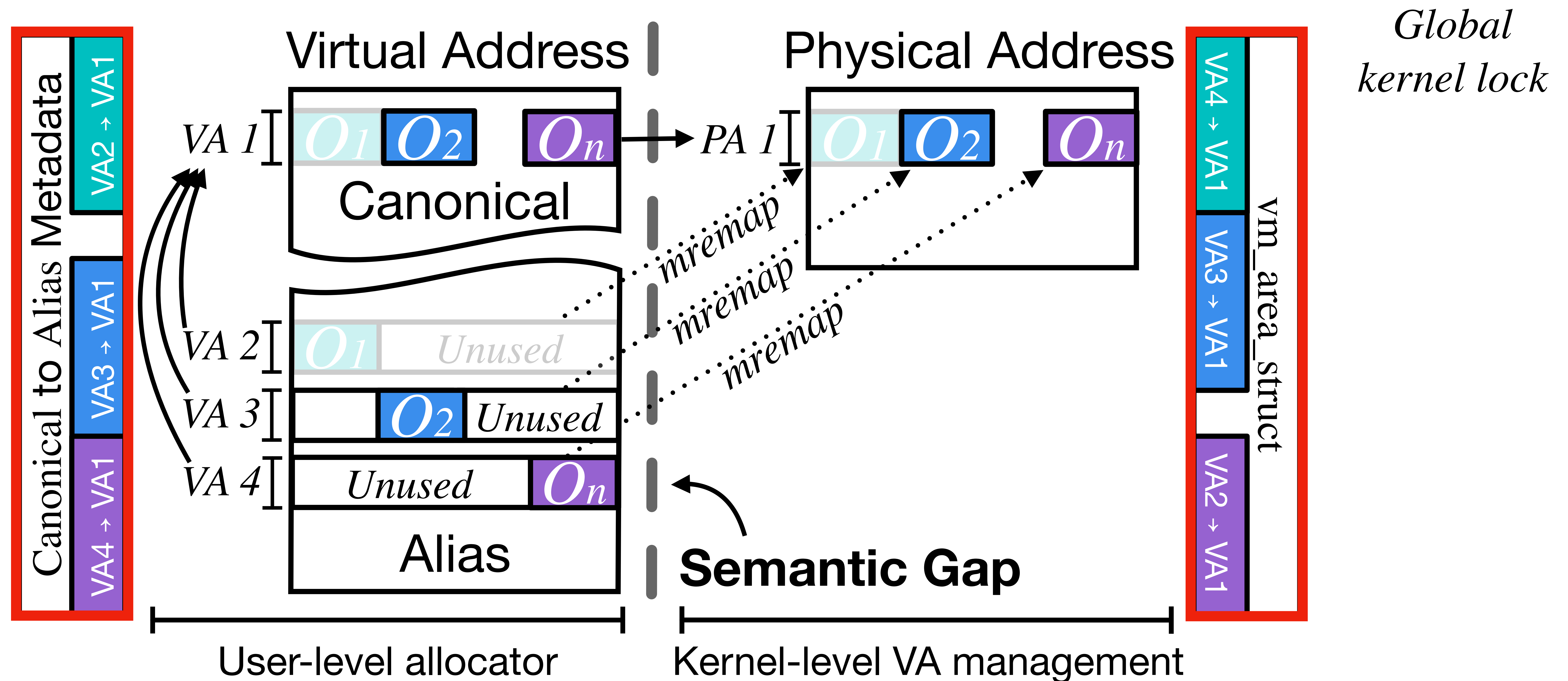


- Kernel manages the virtual address to the physical address mappings
- User and kernel doesn't know each other's states: **Semantic Gap**

KAIST

CASYS Computer Architecture and Systems Lab

Performance Overhead due to Semantic Gap

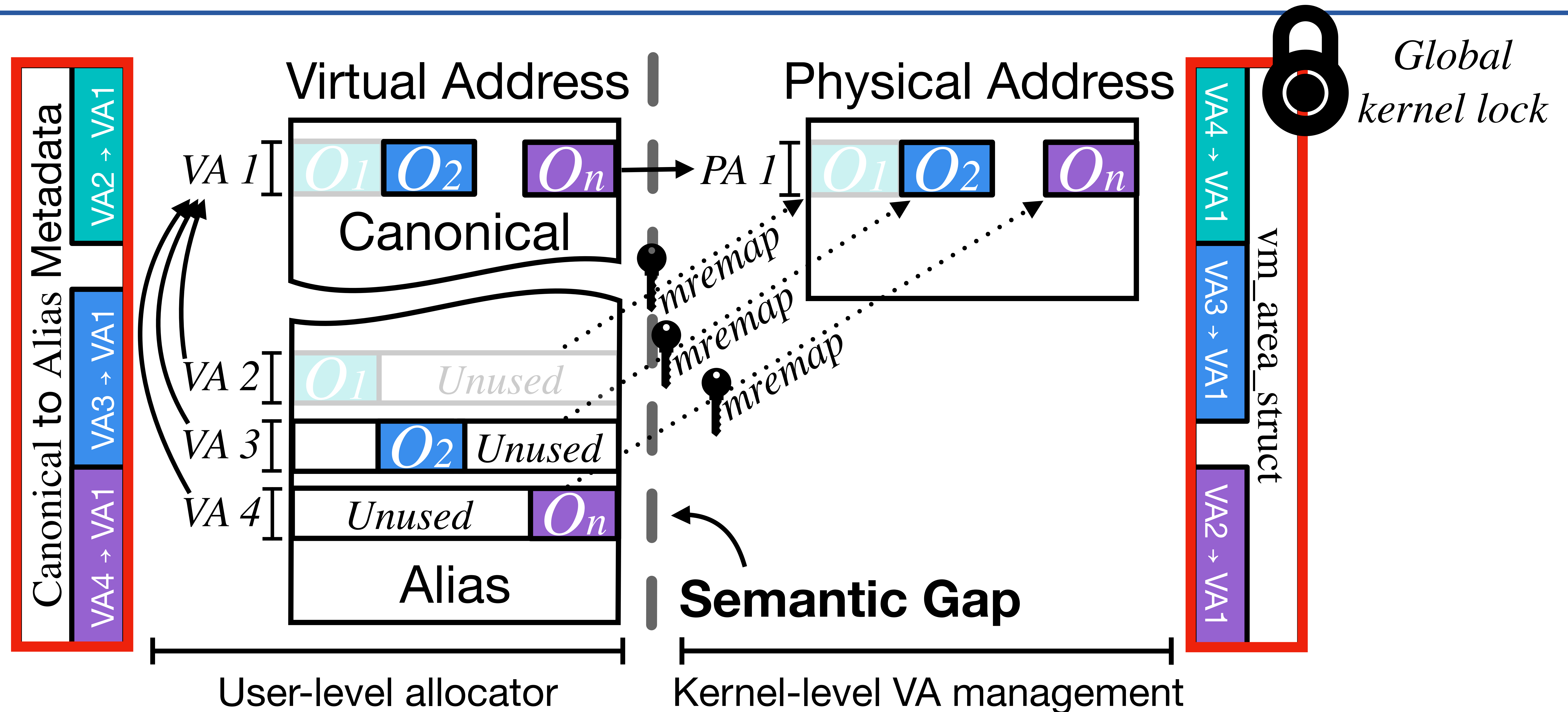


- OTA must issue a costly system call for each allocation to update the kernel
- *Frequent execution of system calls leads to these global locks becoming a major bottleneck*

KAIST

CASYS
Computer Architecture and Systems Lab

Performance Overhead due to Semantic Gap

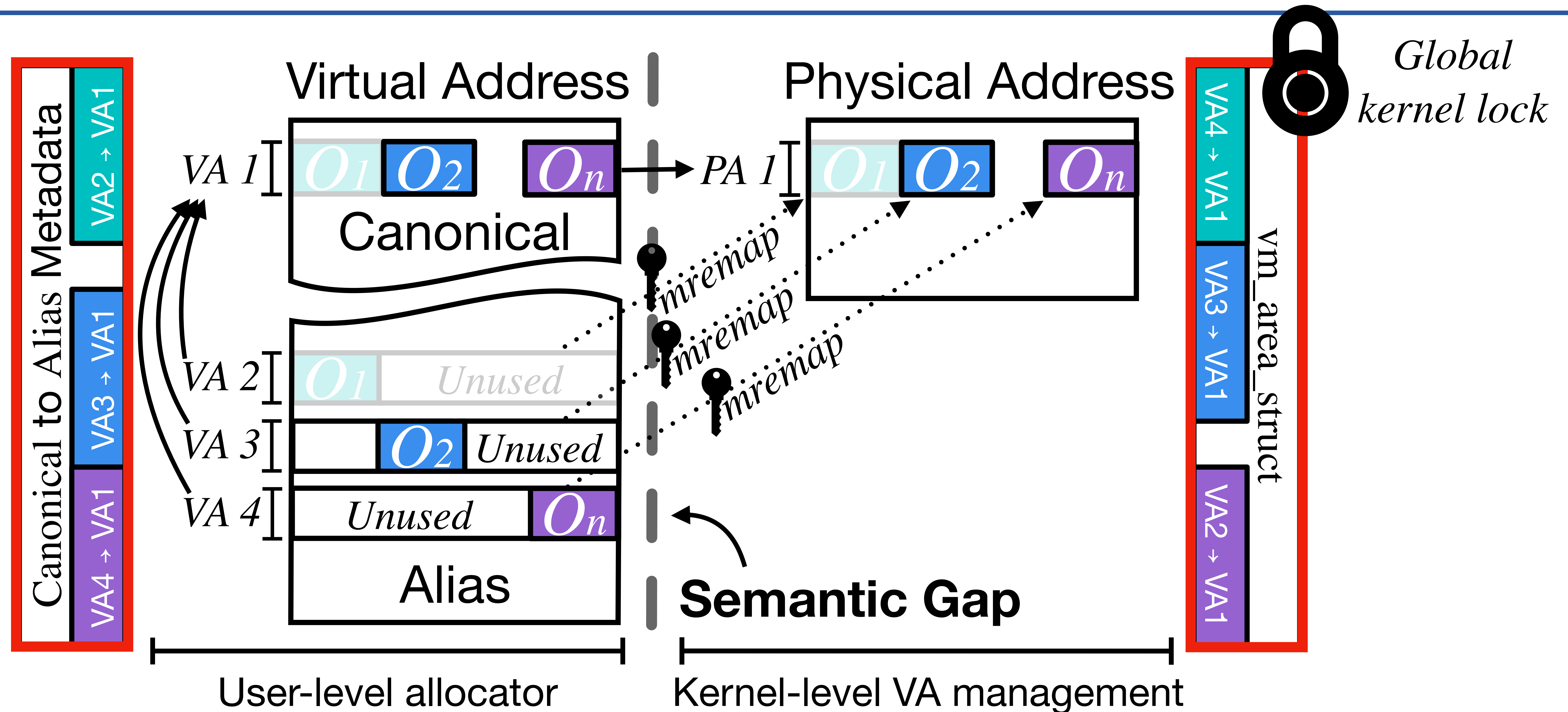


- OTA must issue a costly system call for each allocation to update the kernel
- *Frequent execution of system calls leads to these global locks becoming a major bottleneck*

KAIST

CASYS Computer Architecture and Systems Lab

Performance Overhead due to Semantic Gap



- OTA must issue a costly system call for each allocation to update the kernel

Semantic Gap is the major constraints of OTA

Previous Approaches: How to Address the Semantic Gap

	Oscar	FFMalloc	DangZero
Method	*USENIX Security 2017* Batching system calls	*USENIX Security 2021* Remove alias mapping	*ACM CCS 2022* Library operating system
Pros	+ Moderate memory overhead	+ Fast performance	+ Moderate memory overhead + High bug-detect precision
Cons	- Low performance - Low scalability - Cannot support copy-on-write	- High memory overhead - Low bug-detect precision	- Virtualization overhead - Low scalability - Cannot support copy-on-write - Lack of compatibilities

KAIST

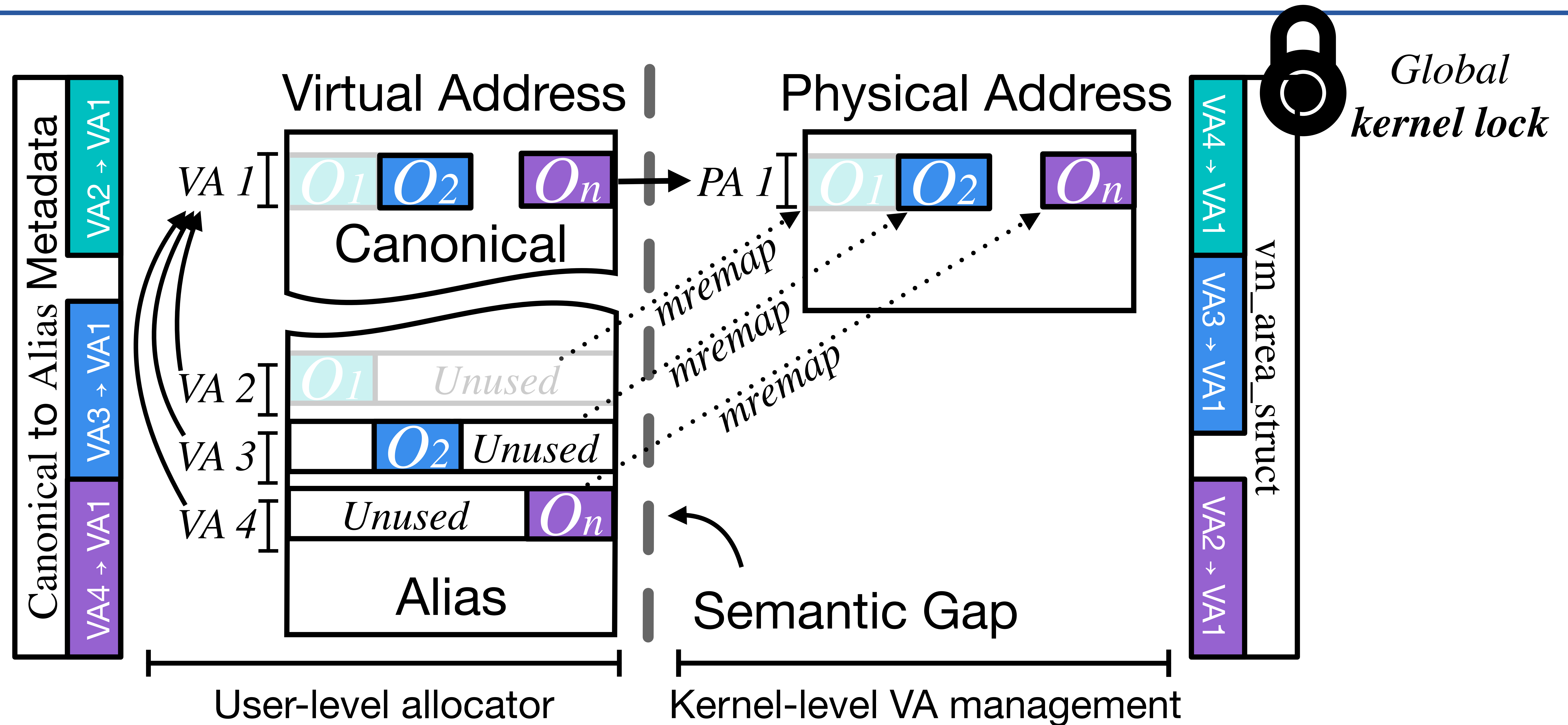
CASYS
Computer Architecture and Systems Lab

Previous Approaches: How to Address the Semantic Gap

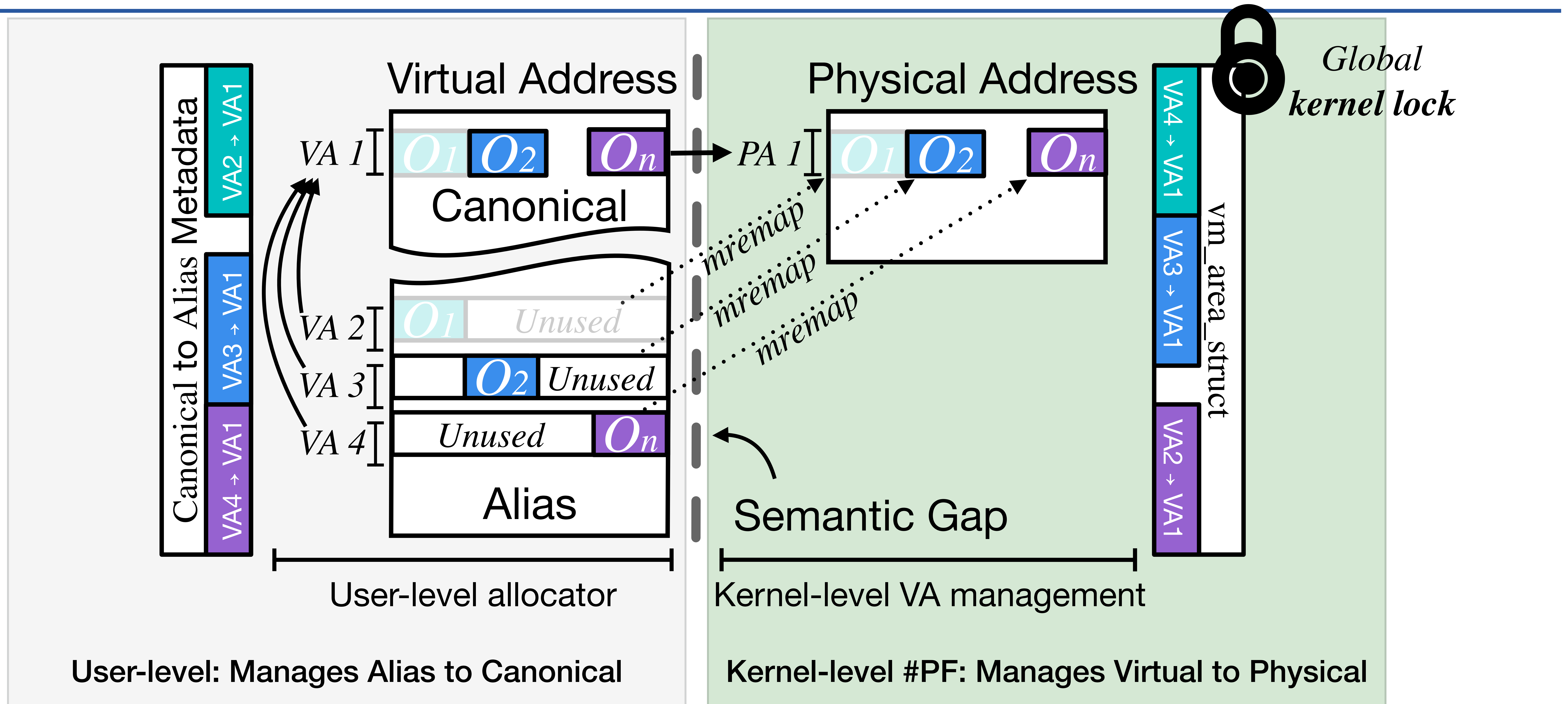
	Oscar	FFMalloc	DangZero
Method	*USENIX Security 2017* Batching system calls	*USENIX Security 2021* Remove alias mapping	*ACM CCS 2022* Library operating system
Pros	+ Moderate memory overhead	+ Fast performance	+ Moderate memory overhead + High bug-detect precision
Cons	- Low performance - Low scalability - Cannot support copy-on-write	- High memory overhead - Low bug-detect precision	- Virtualization overhead - Low scalability - Cannot support copy-on-write - Lack of compatibilities

None of the previous work fully balance performance, memory, bug detection, and compatibility

Previous OTA Design Using Alias Mapping



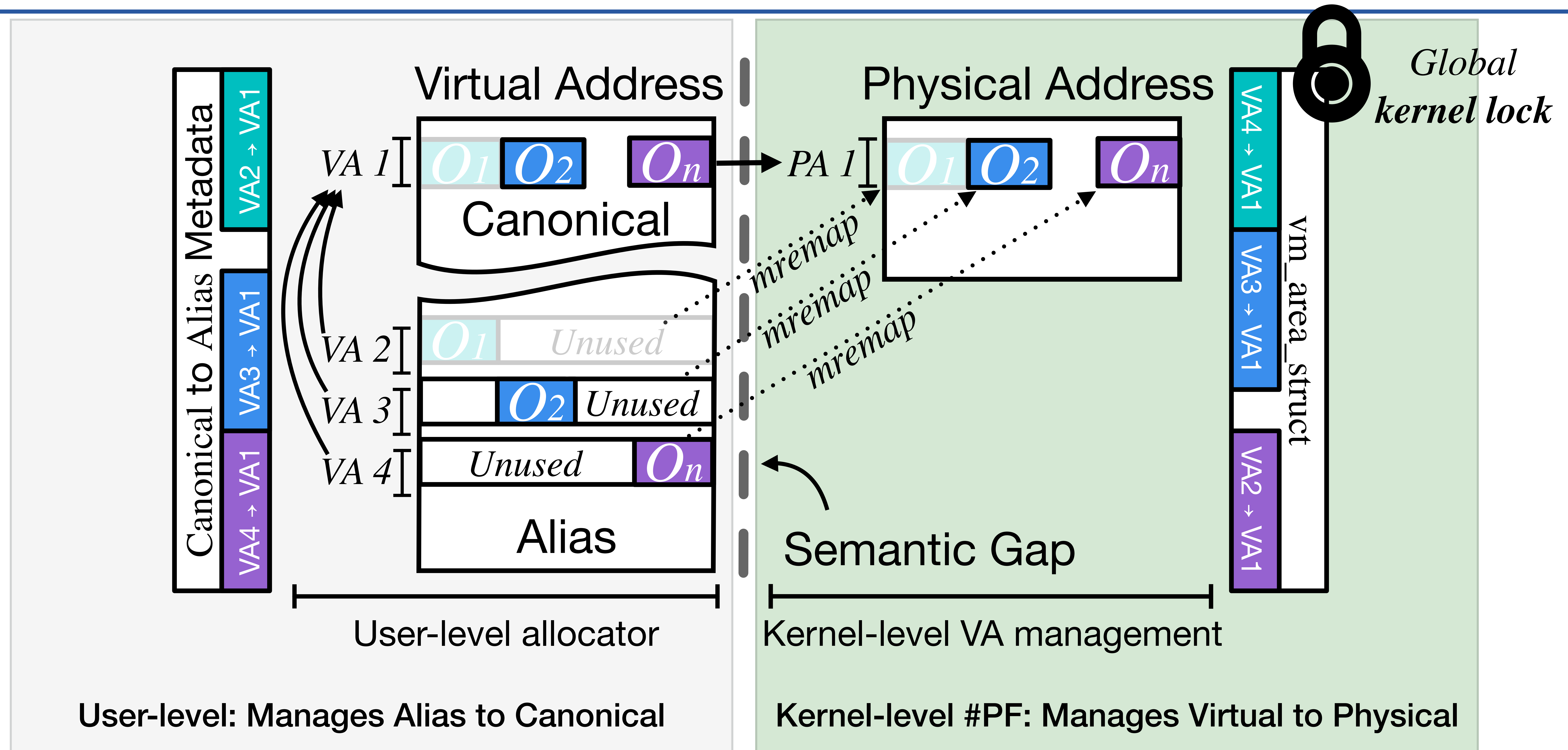
Previous OTA Design Using Alias Mapping



KAIST

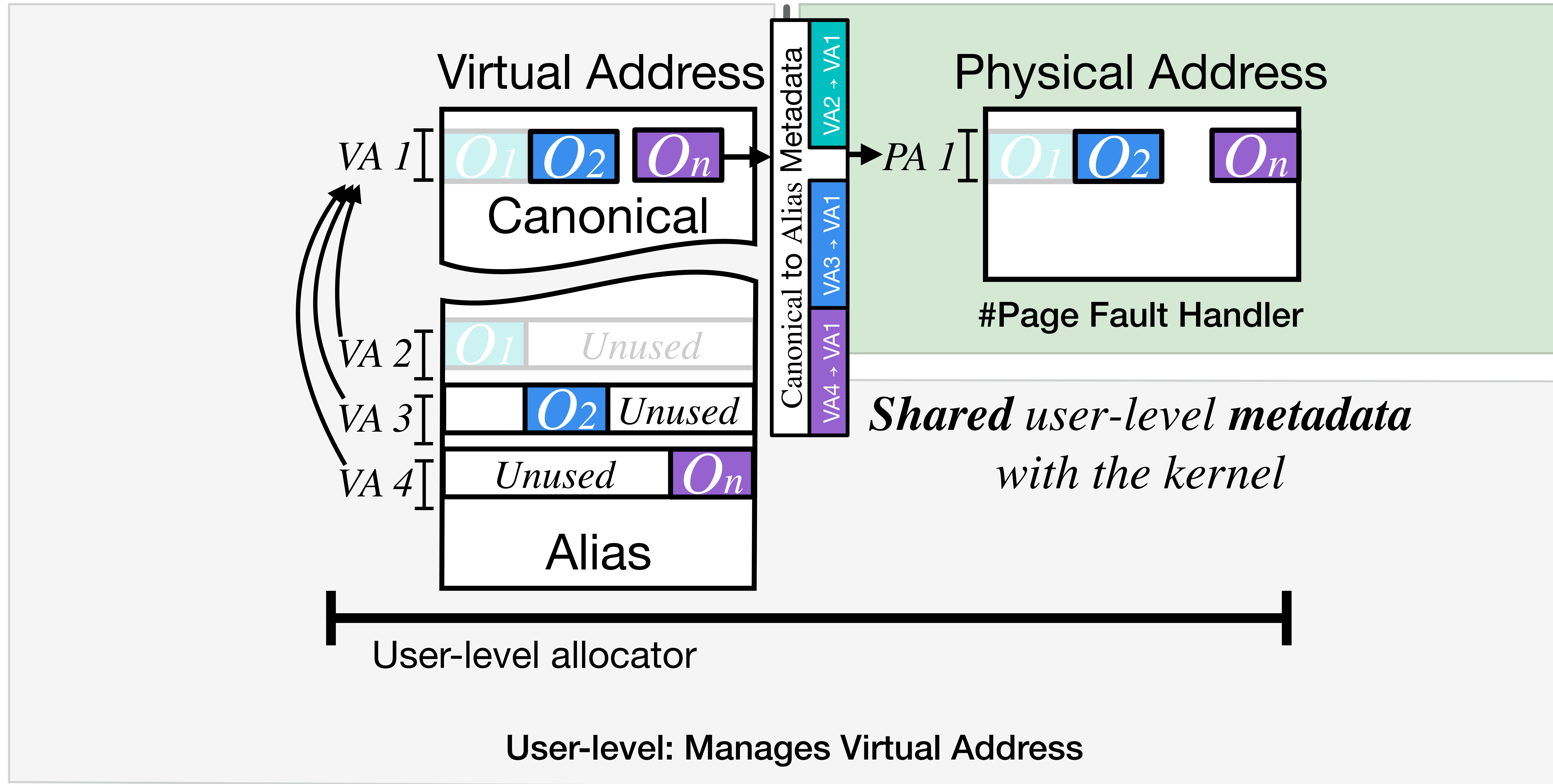
CASYS
Computer Architecture and Systems Lab

Previous OTA Design Using Alias Mapping



Decoupling virtual address management from kernel

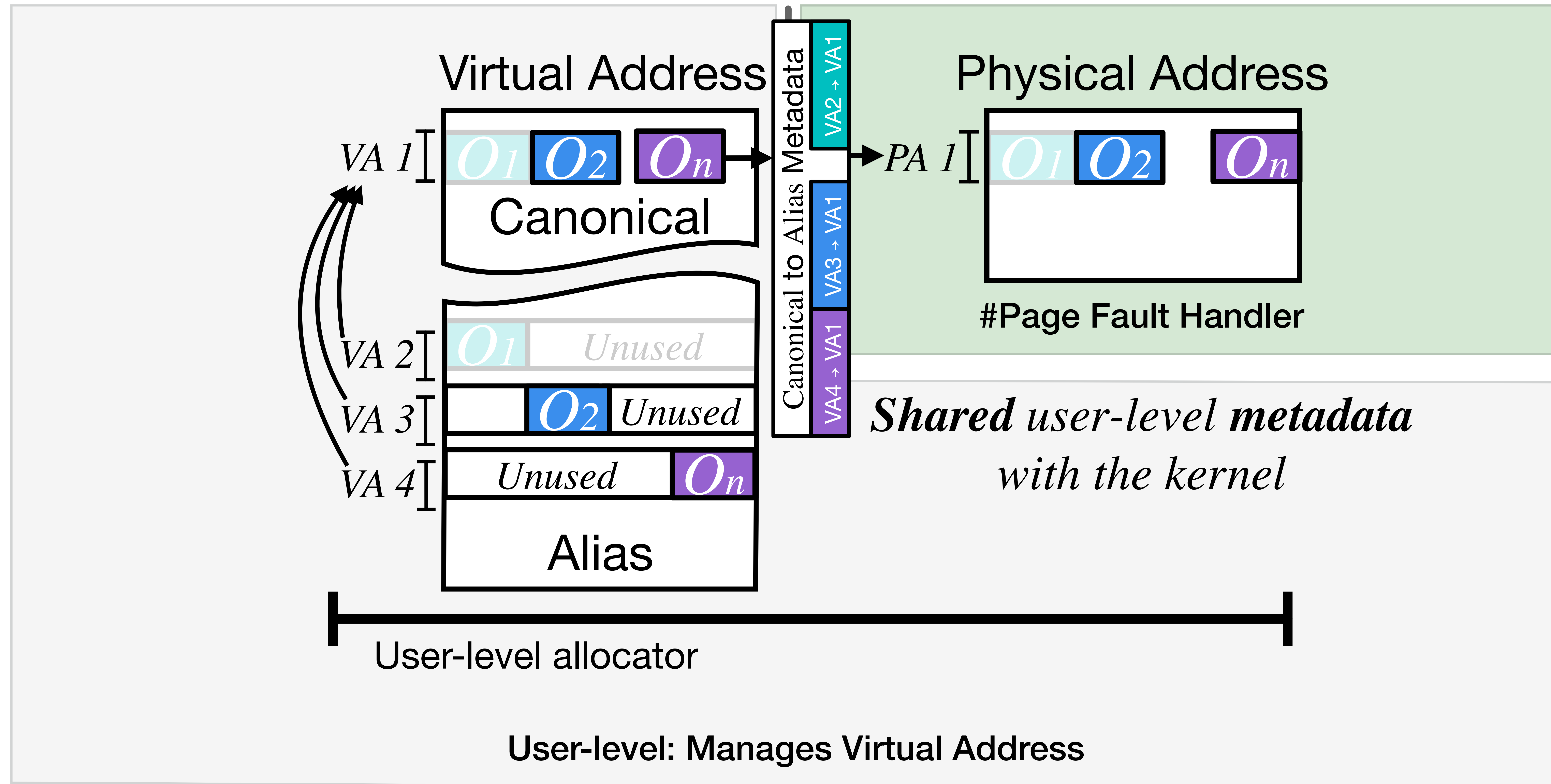
BUDAlloc: Decoupling Virtual Address Management from Kernel



KAIST

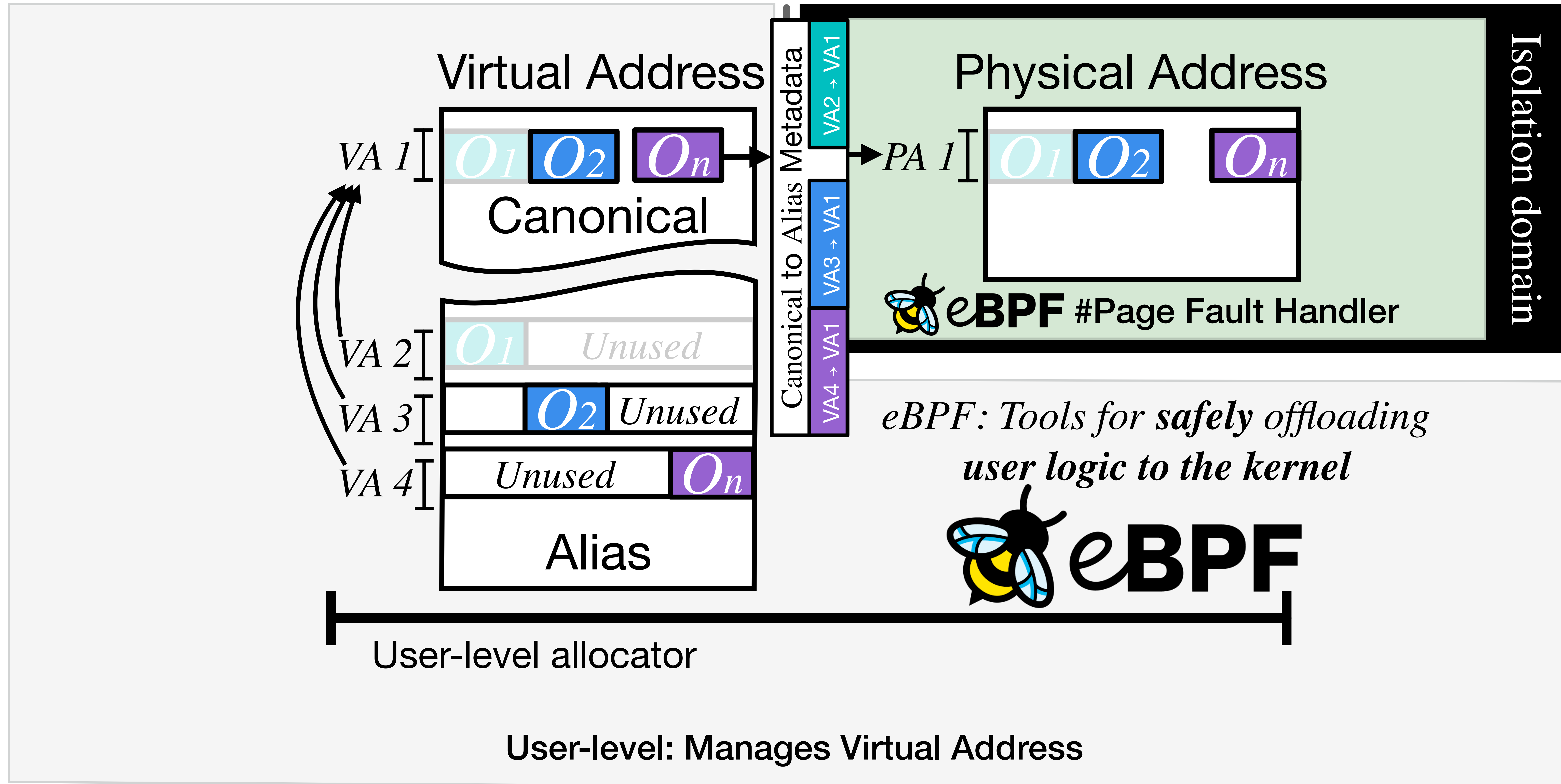
CASYS Computer Architecture and Systems Lab

BUDAlloc: Decoupling Virtual Address Management from Kernel



Manages the virtual address directly using a shared metadata

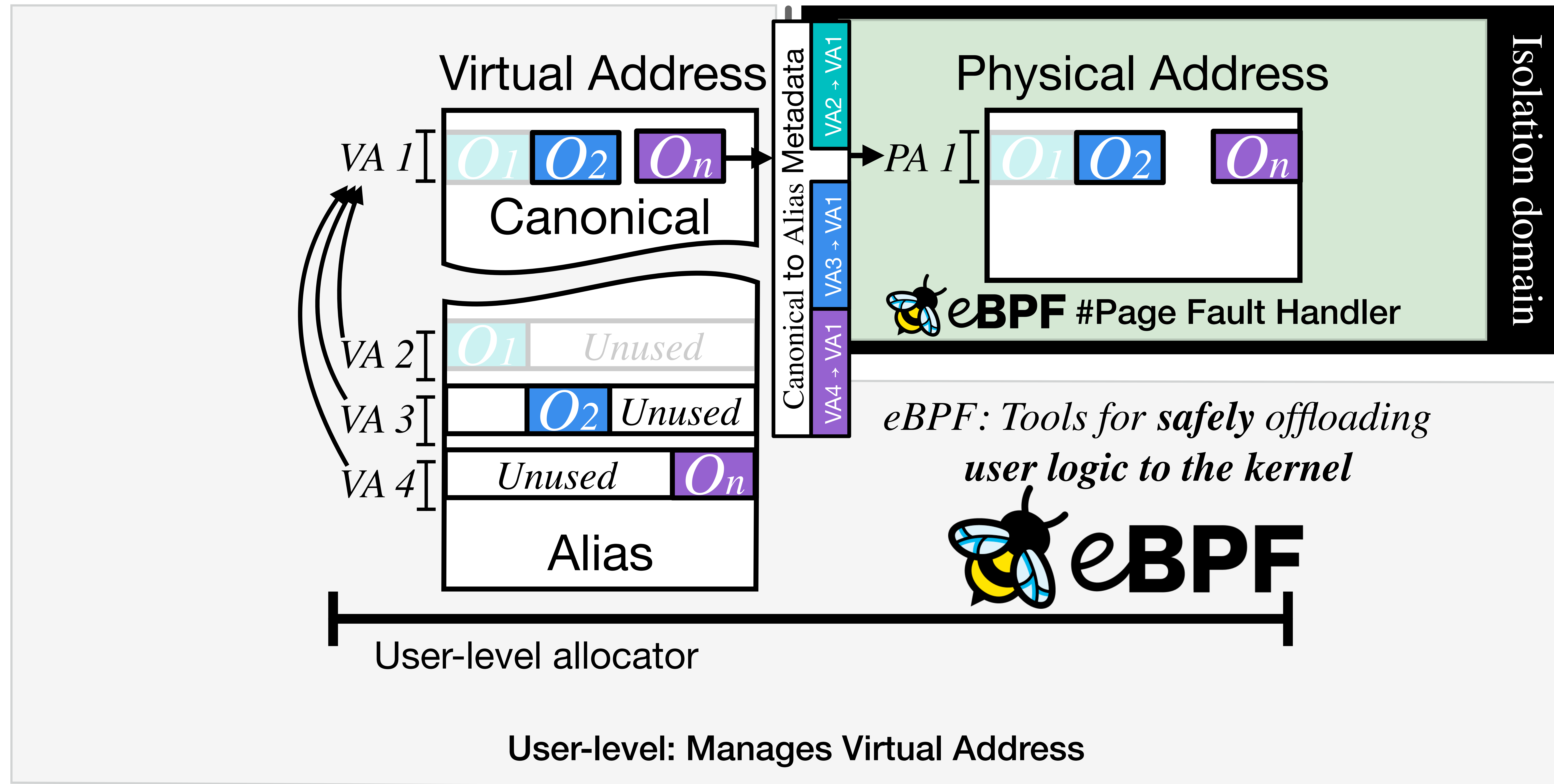
BUDAlloc: Decoupling Virtual Address Management from Kernel



KAIST

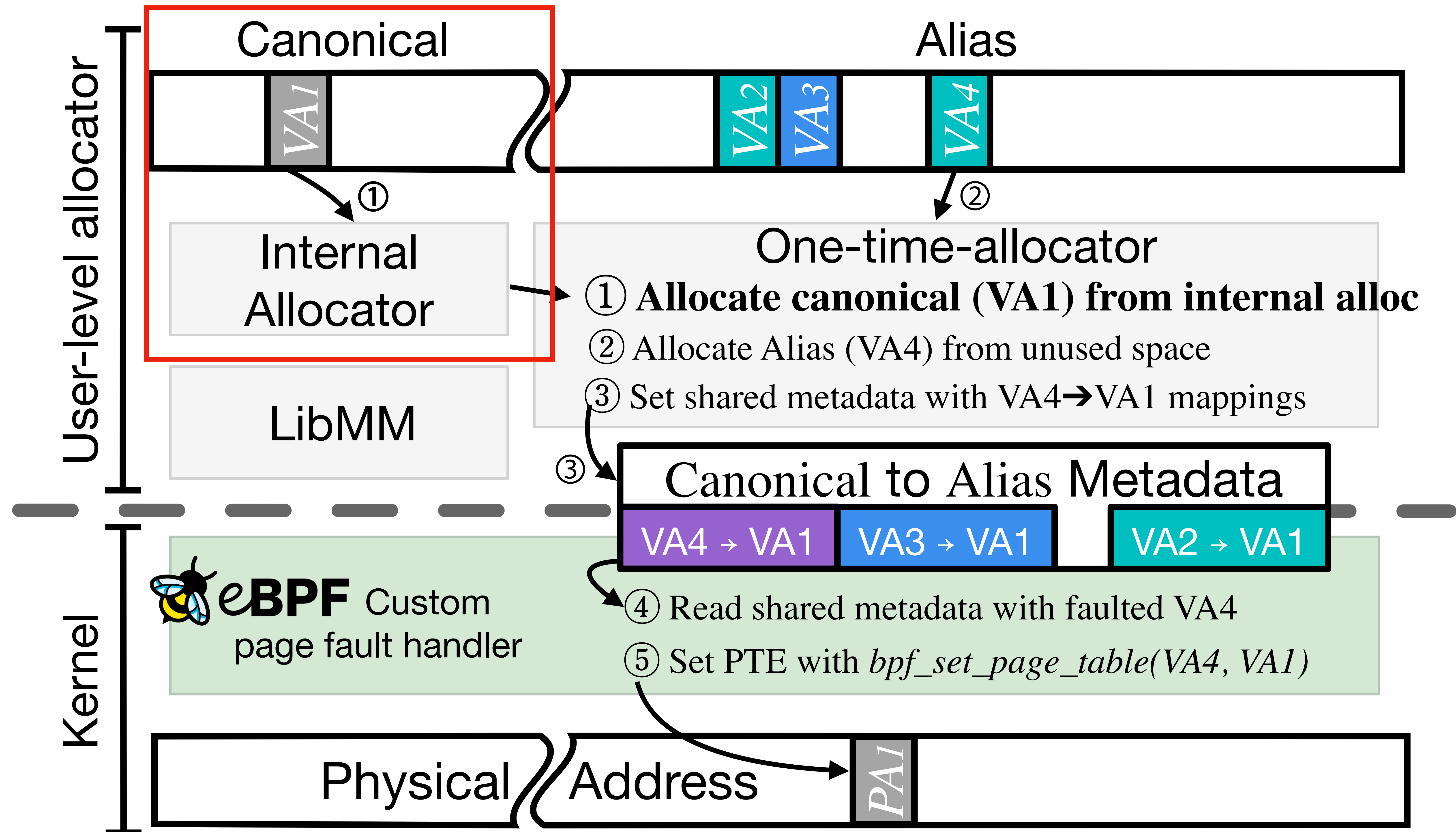
CASYS
Computer Architecture and Systems Lab

BUDAlloc: Decoupling Virtual Address Management from Kernel



Secure and efficient page table modification using eBPF

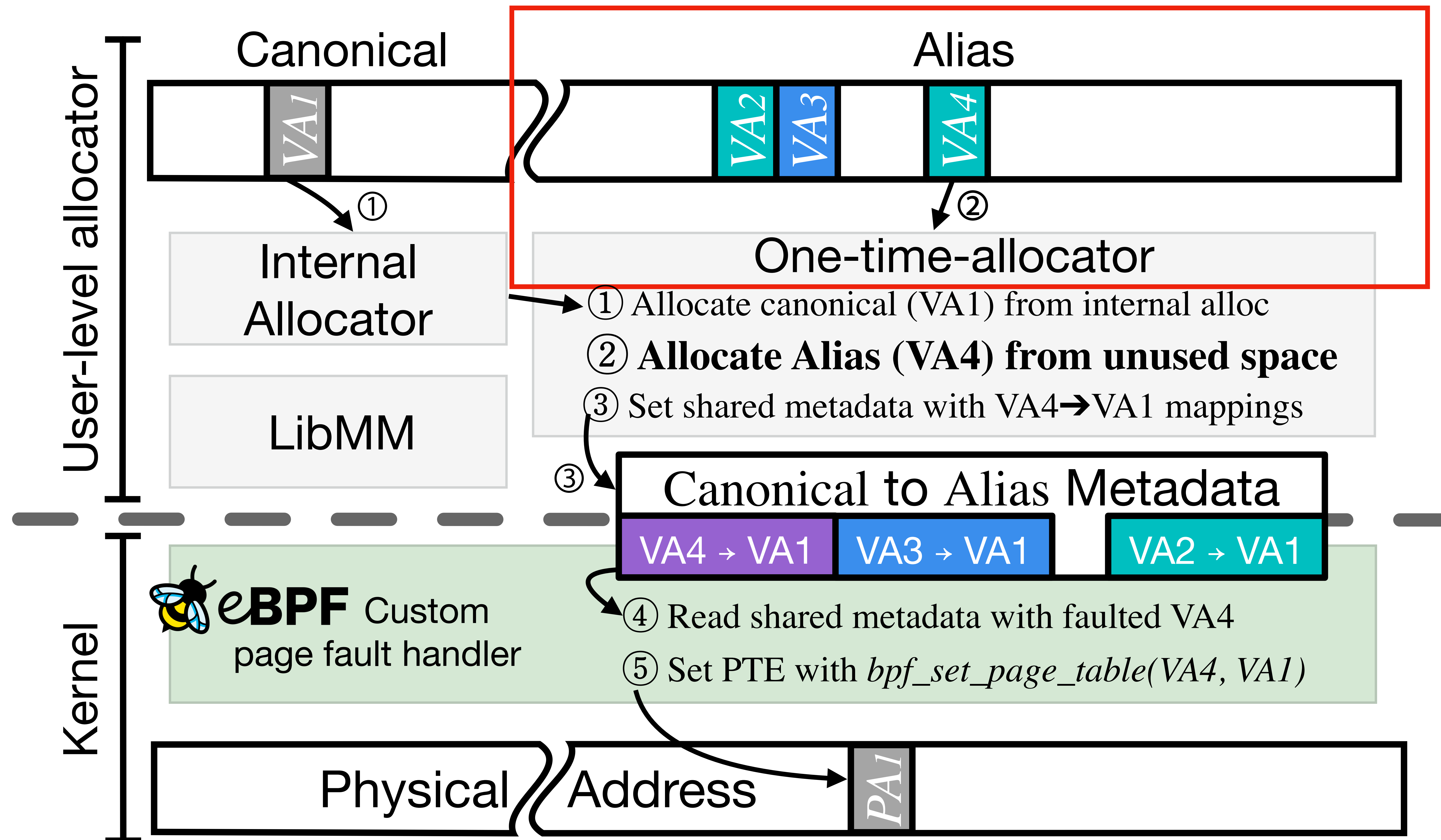
Co-design: User-level Allocator and Kernel #PF Handler



KAIST

CASYS Computer Architecture and Systems Lab

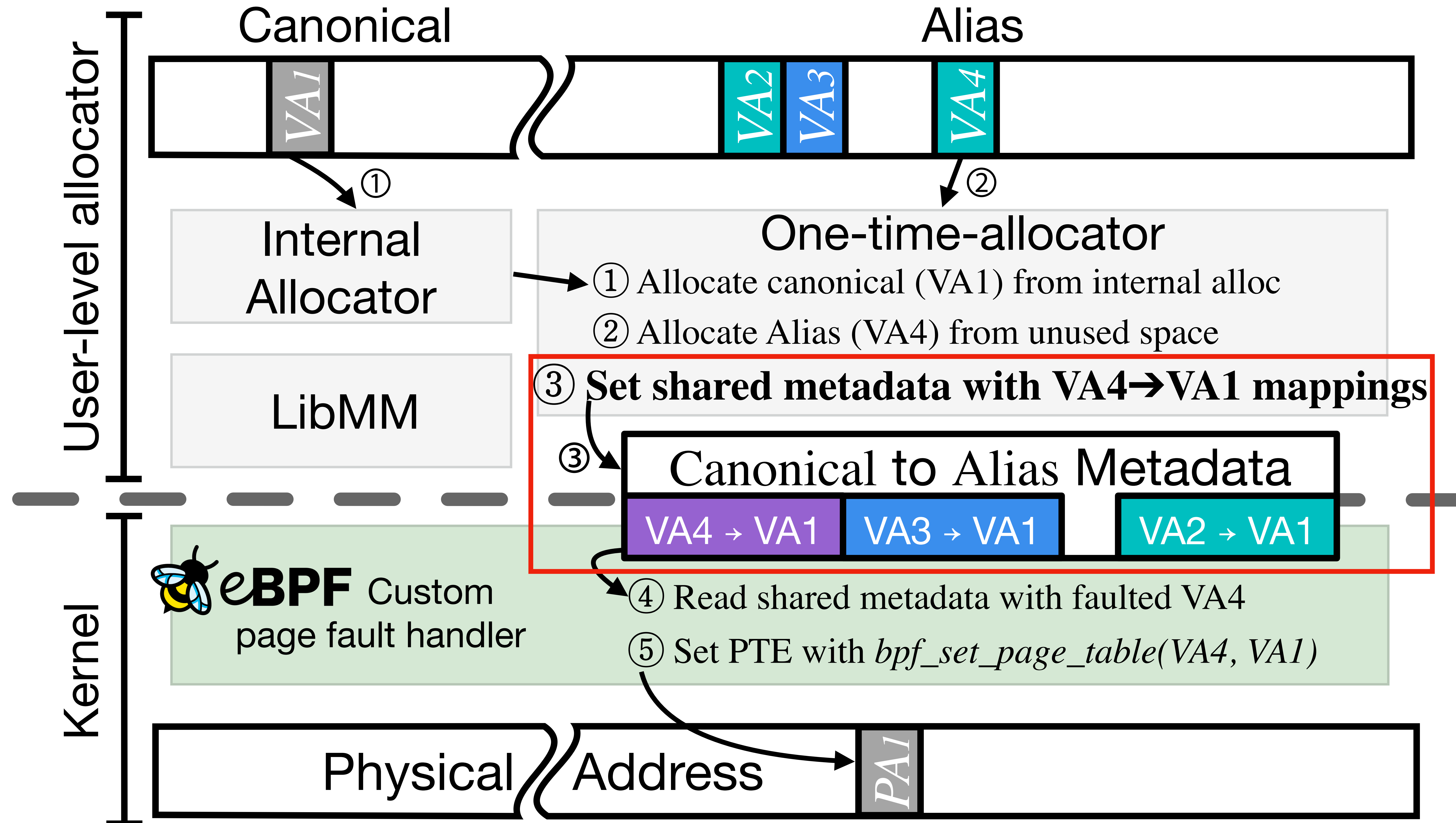
Co-design: User-level Allocator and Kernel #PF Handler



KAIST

CASYS
Computer Architecture and Systems Lab

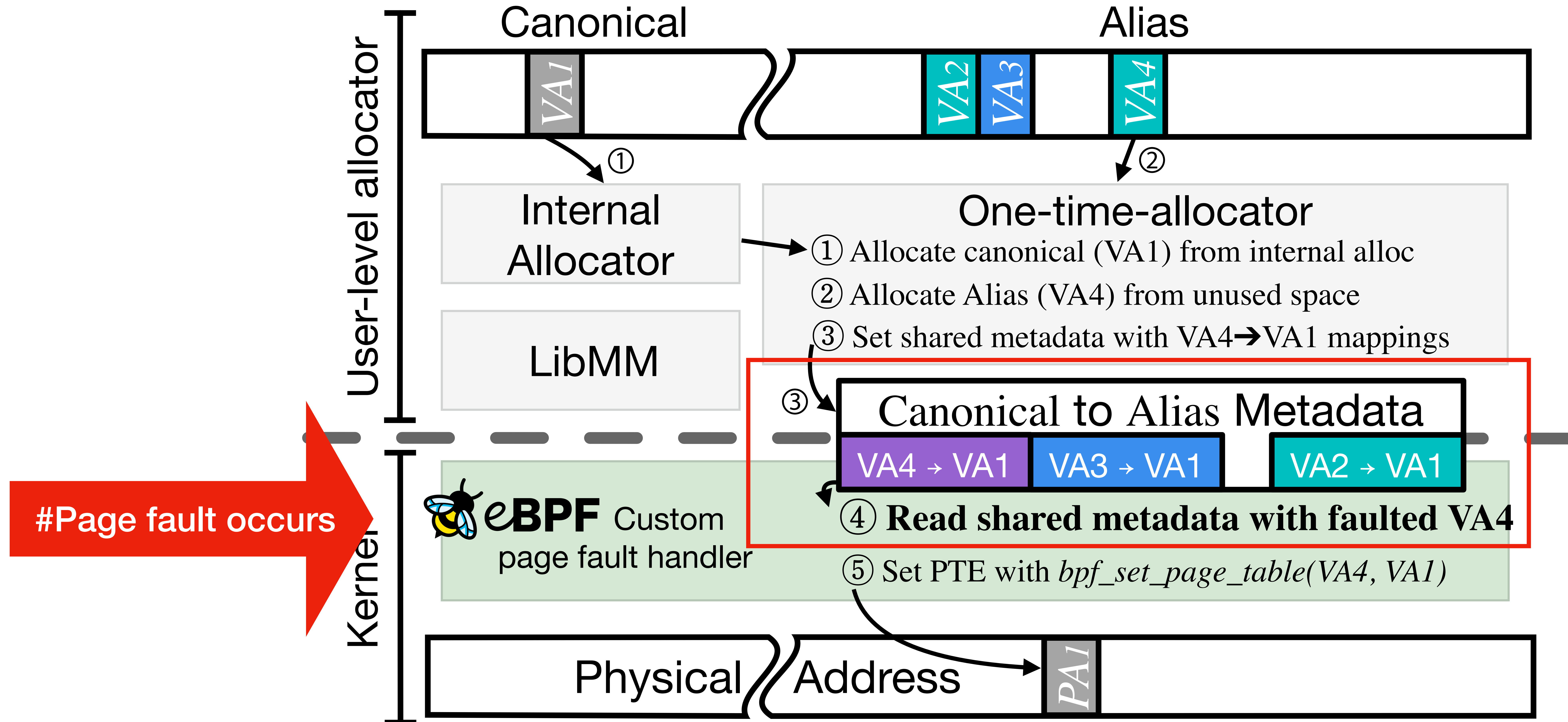
Co-design: User-level Allocator and Kernel #PF Handler



KAIST

CASYS
Computer Architecture and Systems Lab

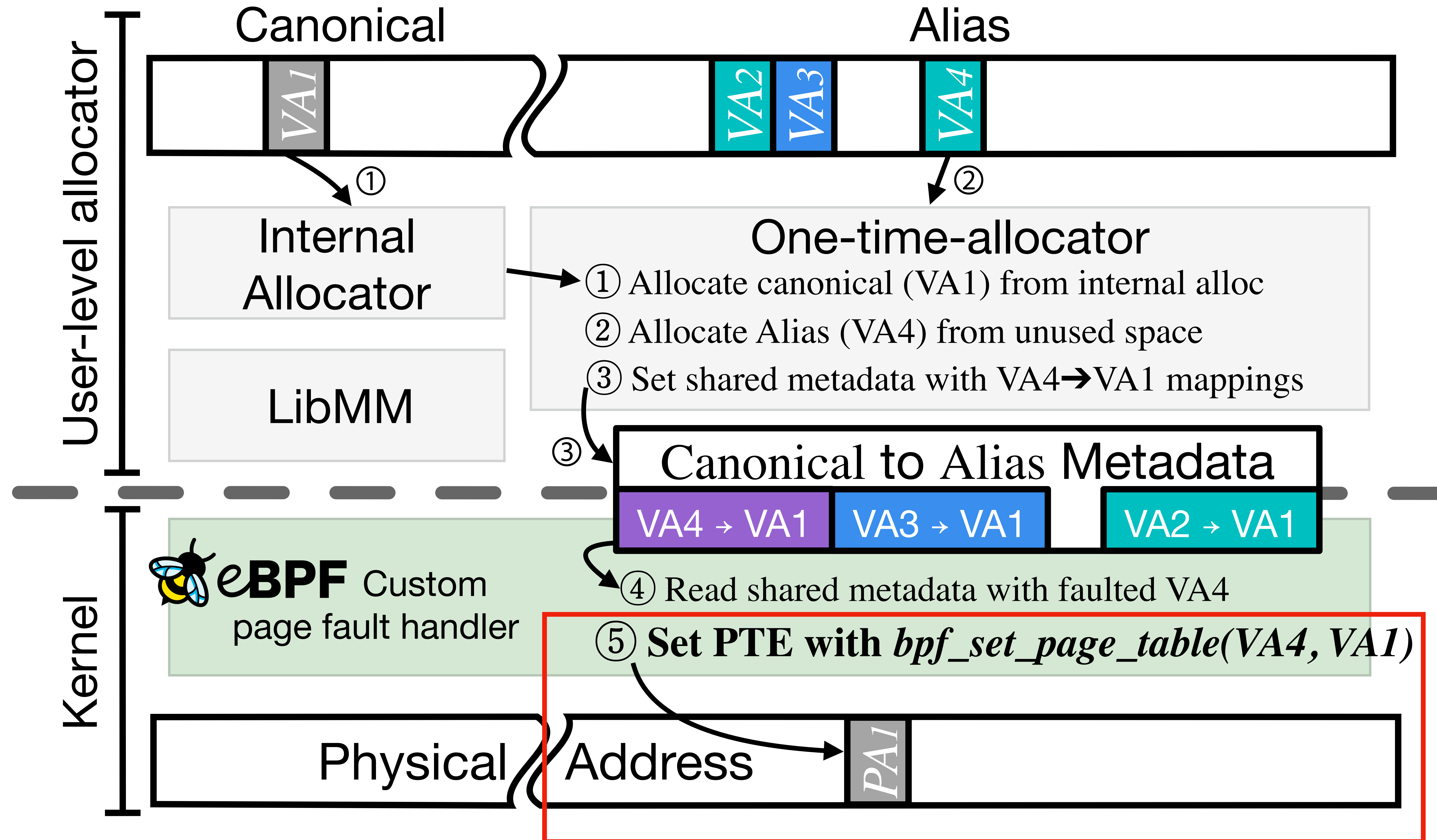
Co-design: User-level Allocator and Kernel #PF Handler



KAIST

CASYS Computer Architecture and Systems Lab

Co-design: User-level Allocator and Kernel #PF Handler



Benefits from the Co-design

- **Performance**
  - BUDAlloc offers an **optional mode** deferring alias address free until the **next page fault,** called **BUDAlloc-prevent**
- **Scalability**
  - BUDAlloc eliminates the global kernel lock
  - BUDAlloc supports **fine-grained locking** using **user-level** semantics
- **Compatibility**
  - BUDAlloc seamlessly **reuses all kernel** physical memory management features

KAIST

CASYS
Computer Architecture and Systems Lab

UAF Bug Detection: BUDAlloc-p Detects Most of Bugs

- BUDAlloc-detection (without deferring)
  - ✓Show **100% precision** of detection
- BUDAlloc-prevention (deferring)
  - ✓Show **97% precision** of detection
- BUDAlloc's detection window is at **least next page fault**

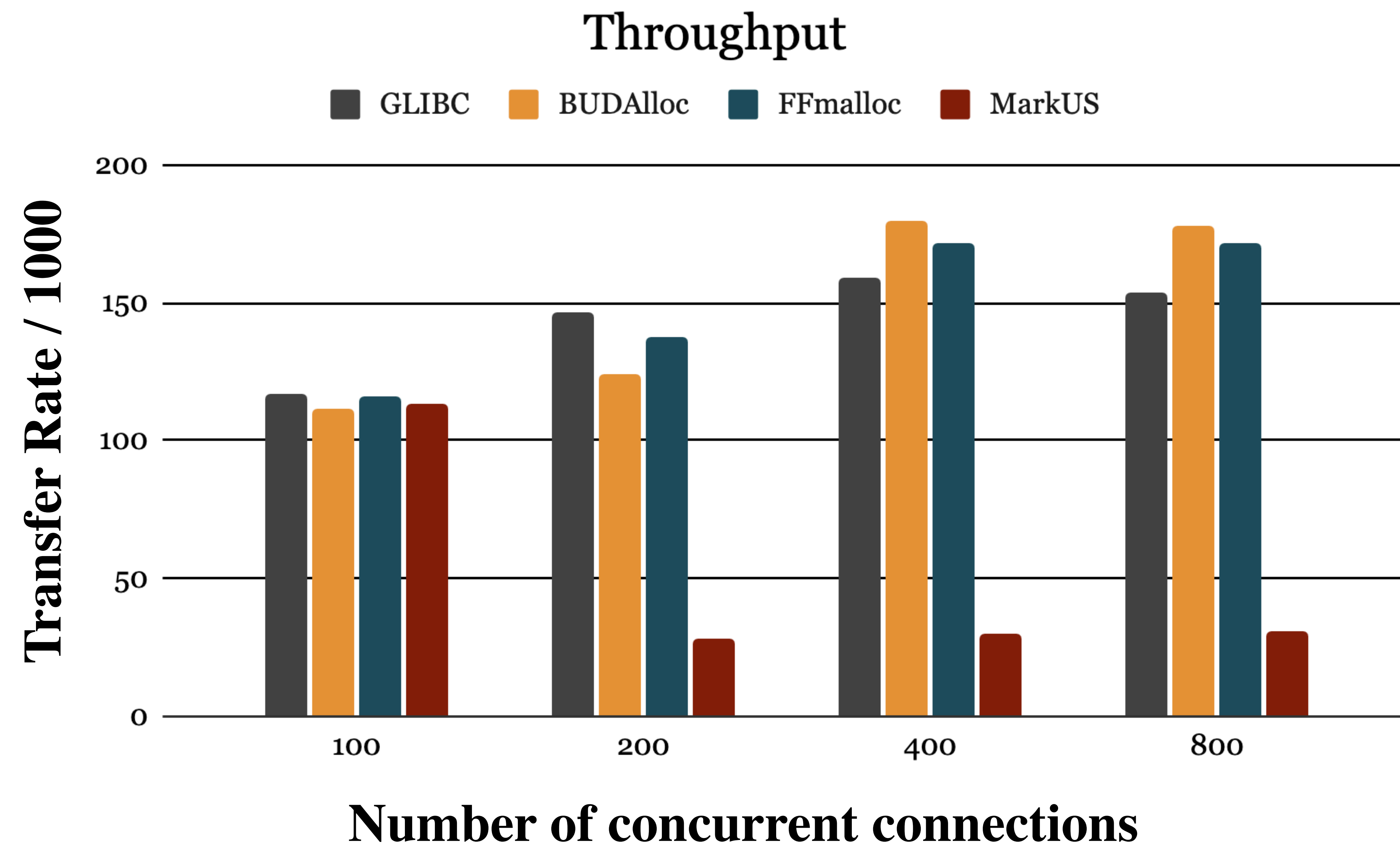
Vulnerability	Program	BUDAlloc-p	BUDAlloc-d	FFmalloc	DangZero
UAFBench					
CVE-2016-3189	bzip2	●	●	◑	●
*CVE-2016-4487	cxxfilt	●	●	●	●
CVE-2017-10686	nasm	●	●	◑	●
CVE-2018-10685	lrzip	●	●	◑	●
CVE-2018-11496	lrzip	●	●	◑	●
*CVE-2018-11416	jpegoptim	●	●	●	●
CVE-2018-20623	readelf	●	●	◑	●
*CVE-2019-20633	patch	●	●	●	●
*CVE-2019-6455	rec2csv	●	●	●	●
Issue 74	giflib	●	●	◑	●
*Issue 122	gifsicle	●	●	●	●
Issue 73	mjs	●	●	◑	●
Issue 78	mjs	●	●	◑	●
Issue 91	yasm	●	●	◑	●
ffmalloc & DangZero					
CVE-2015-2787	PHP	●	●	◑	●
*CVE-2015-3205	libmimedir	●	●	●	●
CVE-2015-6835	PHP	●	●	◑	●
CVE-2016-5773	PHP	●	●	◑	●
Issue 3515	mruby	●	●	◑	●
Issue 24613	Python	●	●	◑	●
Exploit Database					
CVE-2019-6076	Lua	●	●	◑	●
CVE-2019-7703	Binaryen	●	●	◑	●
CVE-2019-8343	nasm	●	●	◑	●
CVE-2019-17582	libzip	●	●	◑	●
CVE-2020-24346	nginx	●	●	◑	●
CVE-2022-1934	mruby	●	●	◑	●
CVE-2022-1106	mruby	◑	●	◑	●
CVE-2022-35164	LibreDWG	●	●	◑	●
*BUG-66783	PHP	●	●	●	●
BUG-80927	PHP	●	●	◑	●

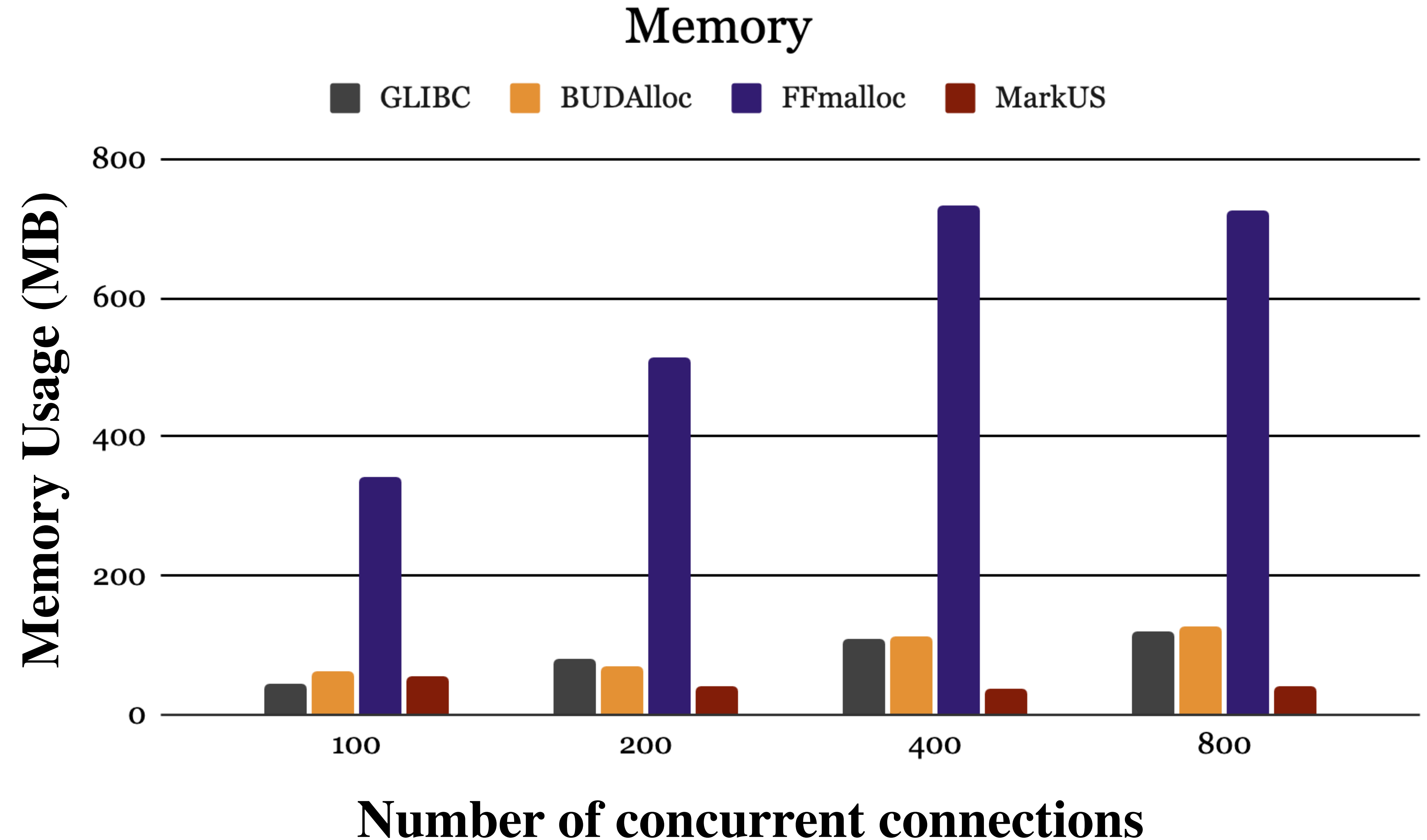
●: Detect UAF bug ◑: Prevent UAF bug

KAIST

CASYS Computer Architecture and Systems Lab

Apache Web Server: BUDAlloc is Fast and Efficient



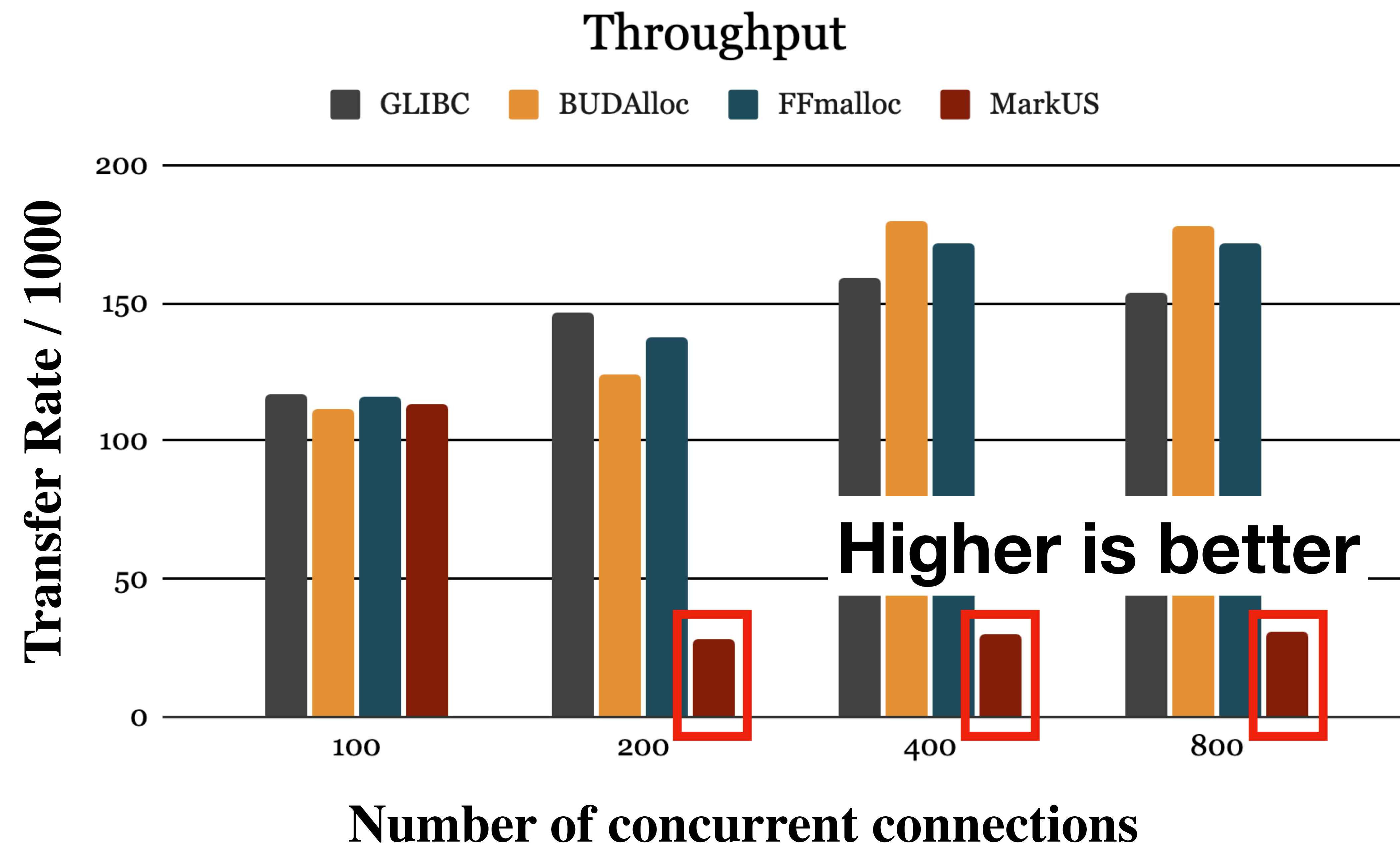


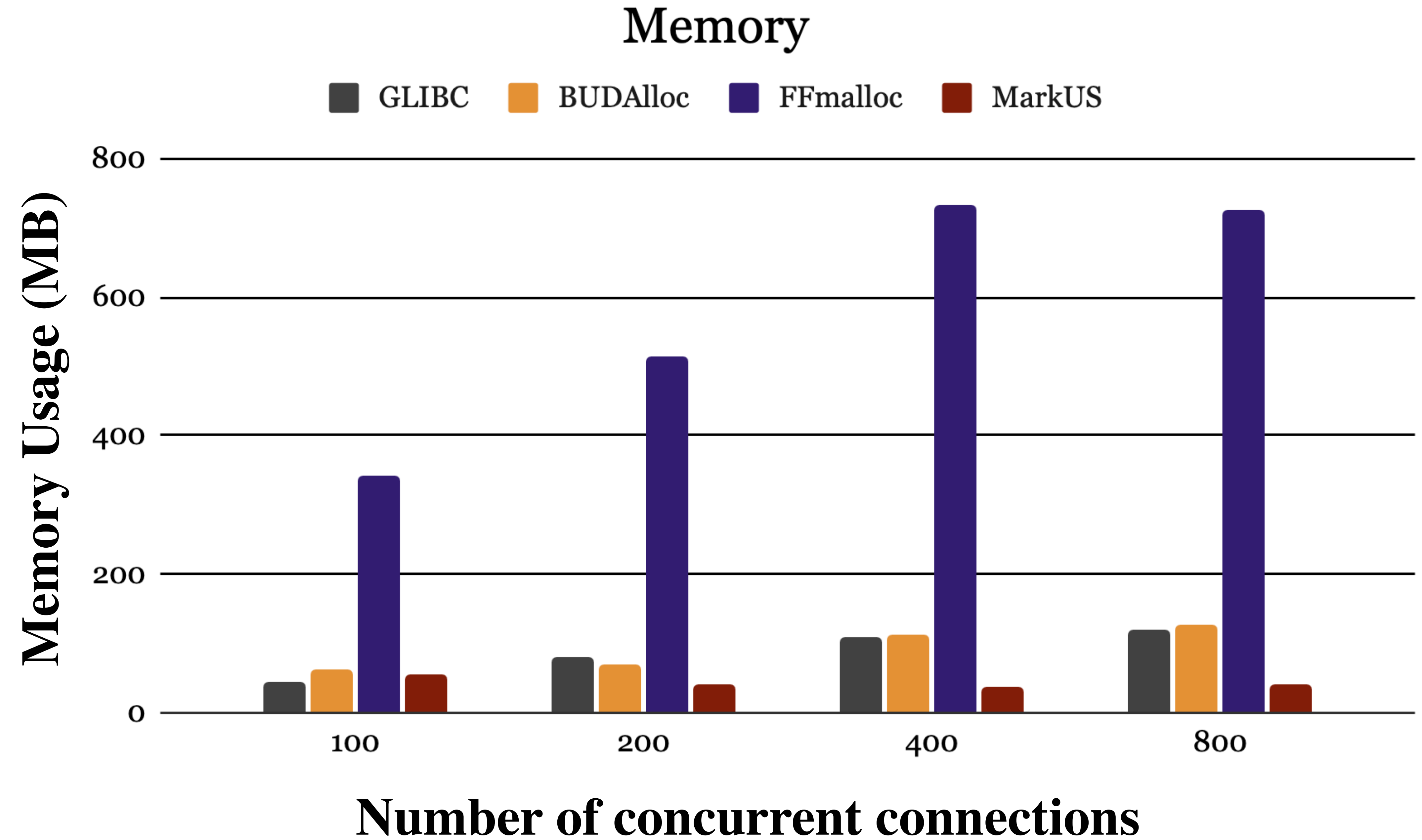
- BUDAlloc shows the **highest scalability** despite the **superior precision of detection**
- BUDAlloc shows the **similar results** with the GLIBC in performance and memory overhead

KAIST

CASYS
Computer Architecture and Systems Lab

Apache Web Server: BUDAlloc is Fast and Efficient



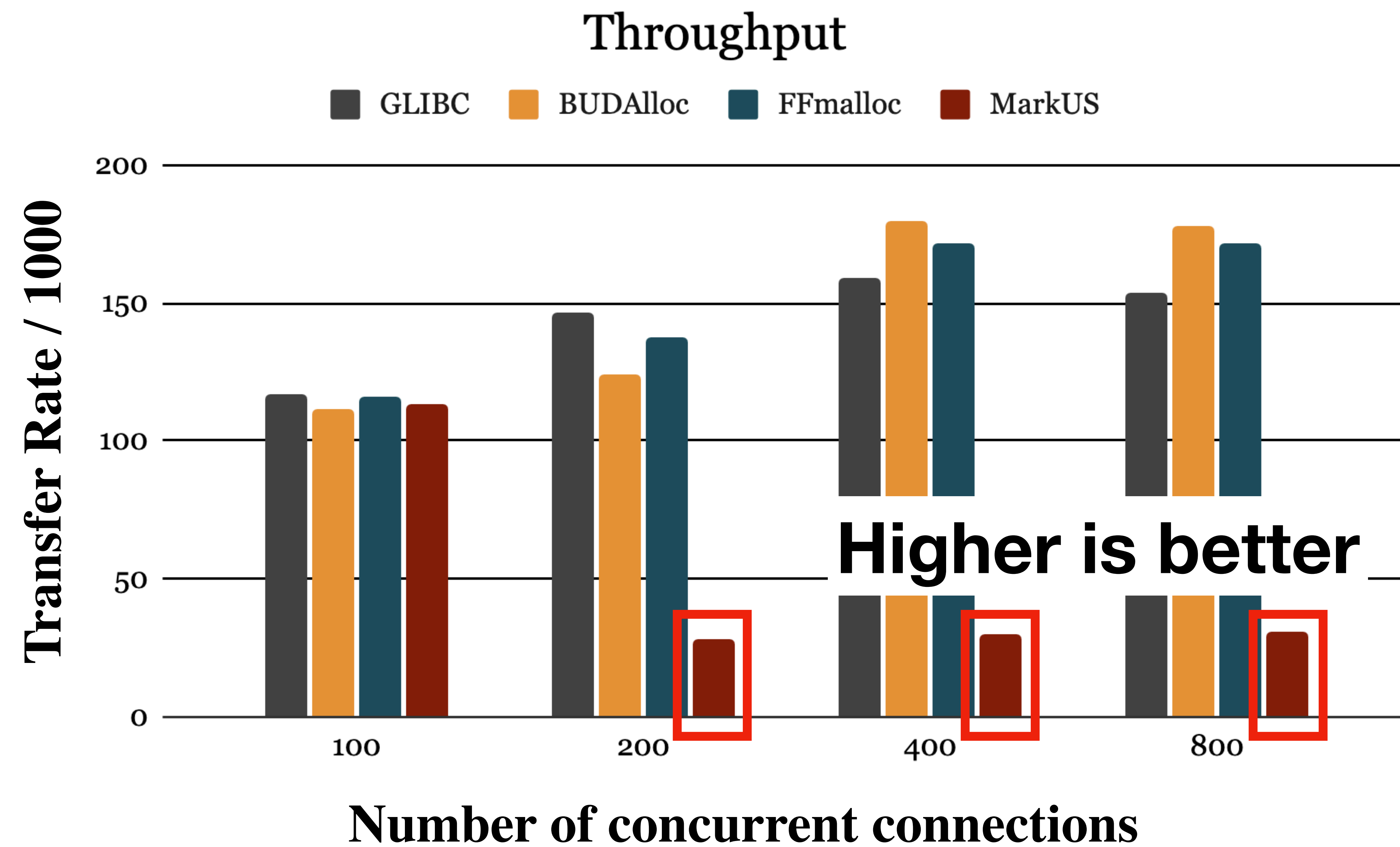


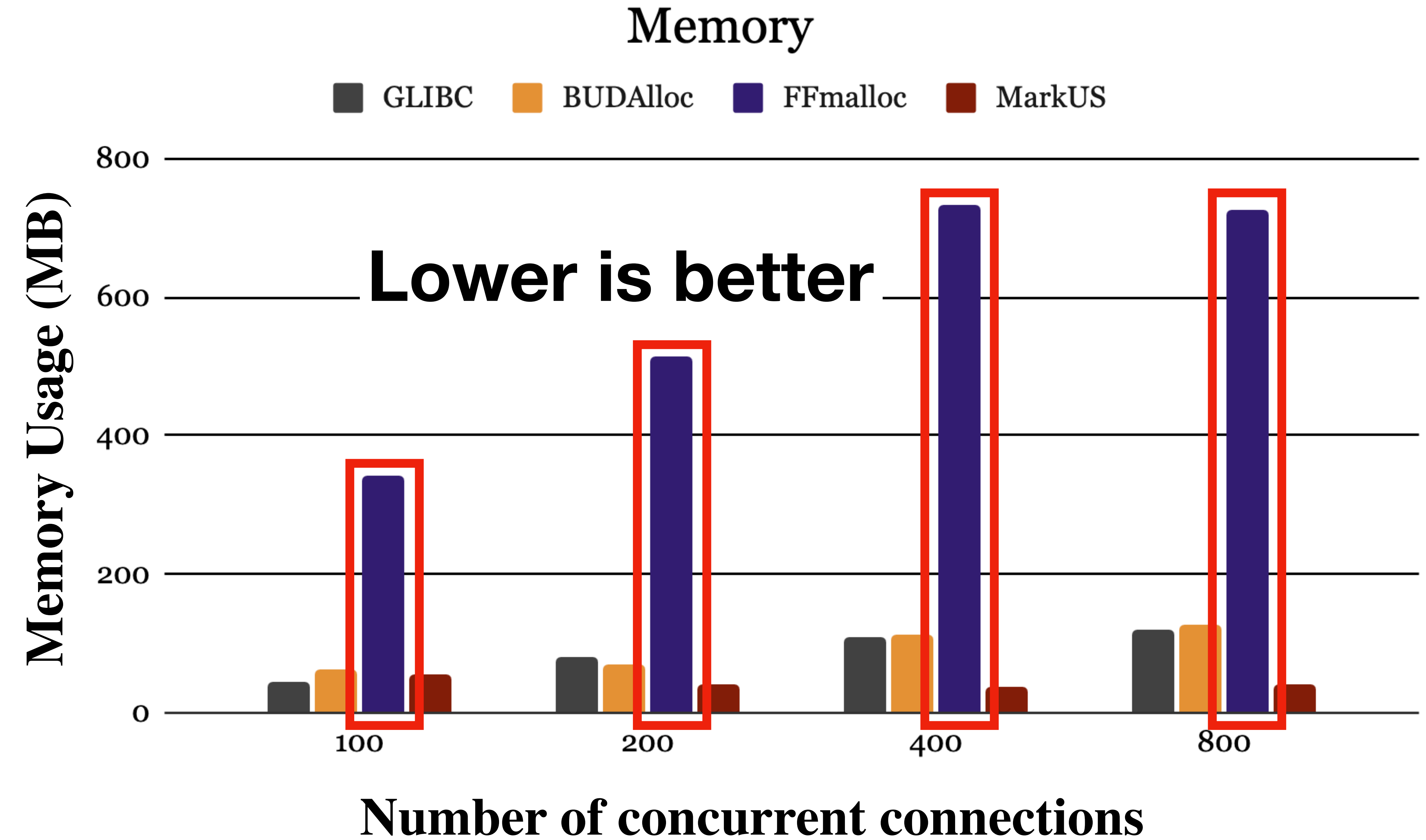
- BUDAlloc shows the **highest scalability** despite the **superior precision of detection**
- BUDAlloc shows the **similar results** with the GLIBC in performance and memory overhead

KAIST

CASYS
Computer Architecture and Systems Lab

Apache Web Server: BUDAlloc is Fast and Efficient



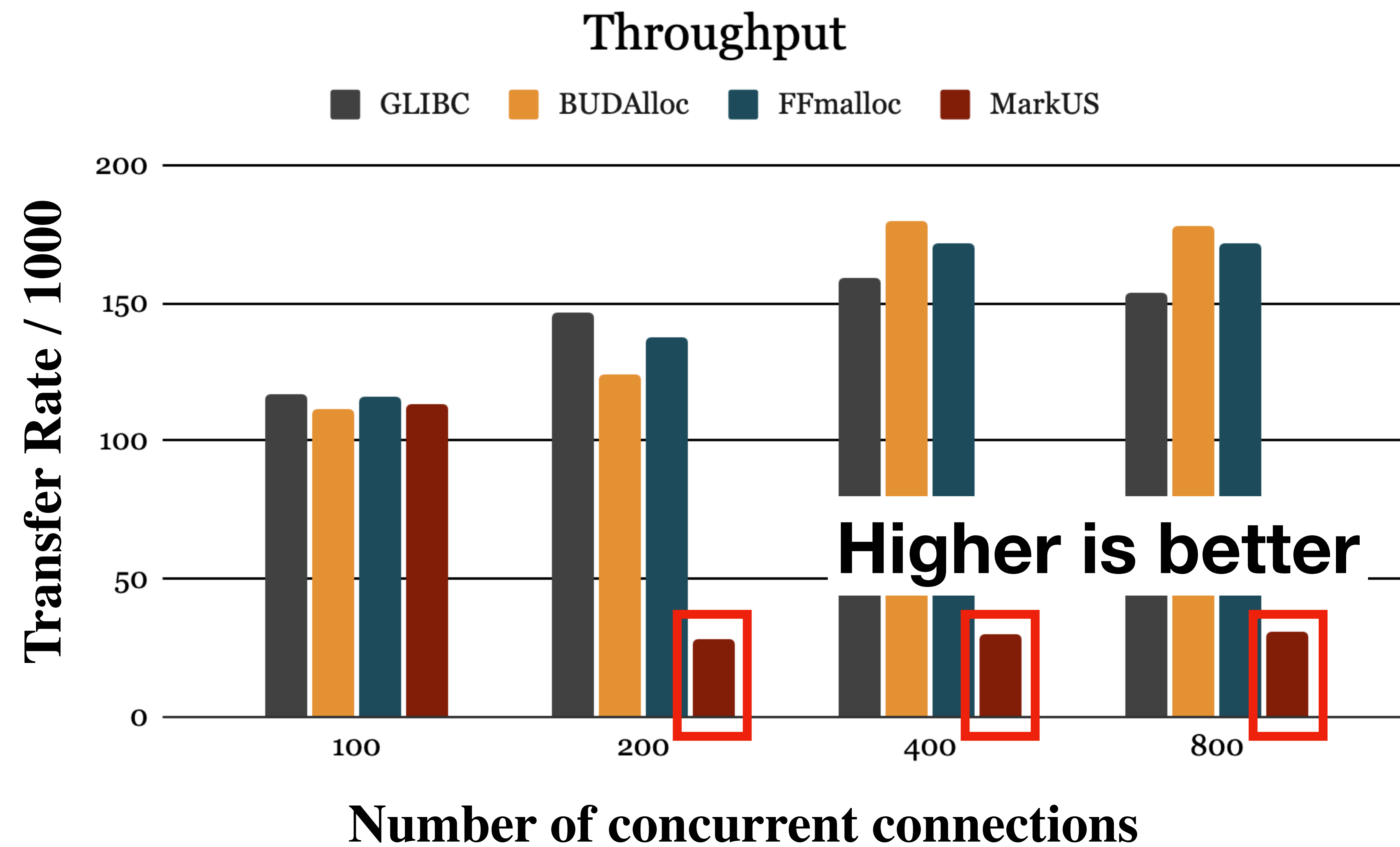


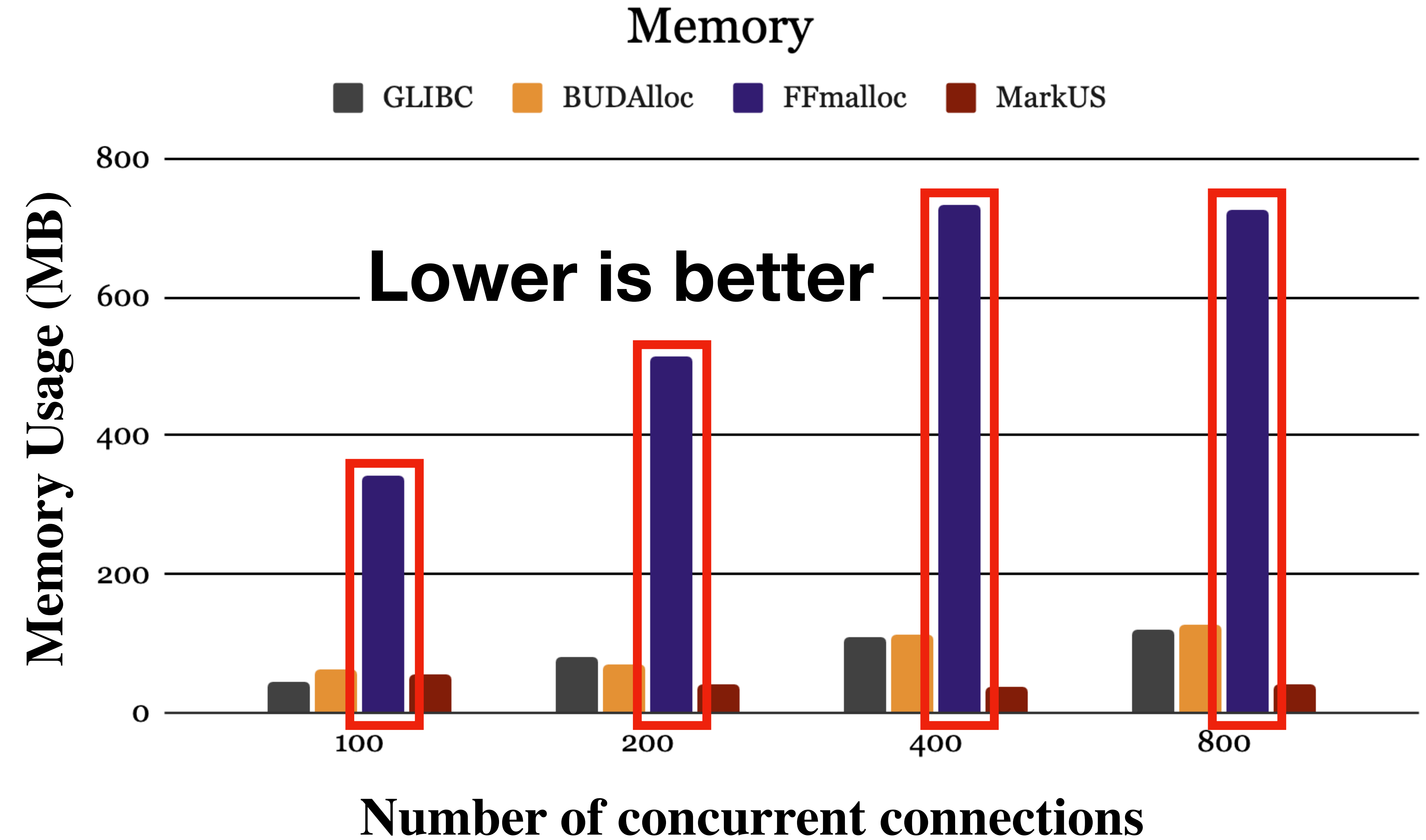
- BUDAlloc shows the **highest scalability** despite the **superior precision of detection**
- BUDAlloc shows the **similar results** with the GLIBC in performance and memory overhead

KAIST

CASYS
Computer Architecture and Systems Lab

Apache Web Server: BUDAlloc is Fast and Efficient





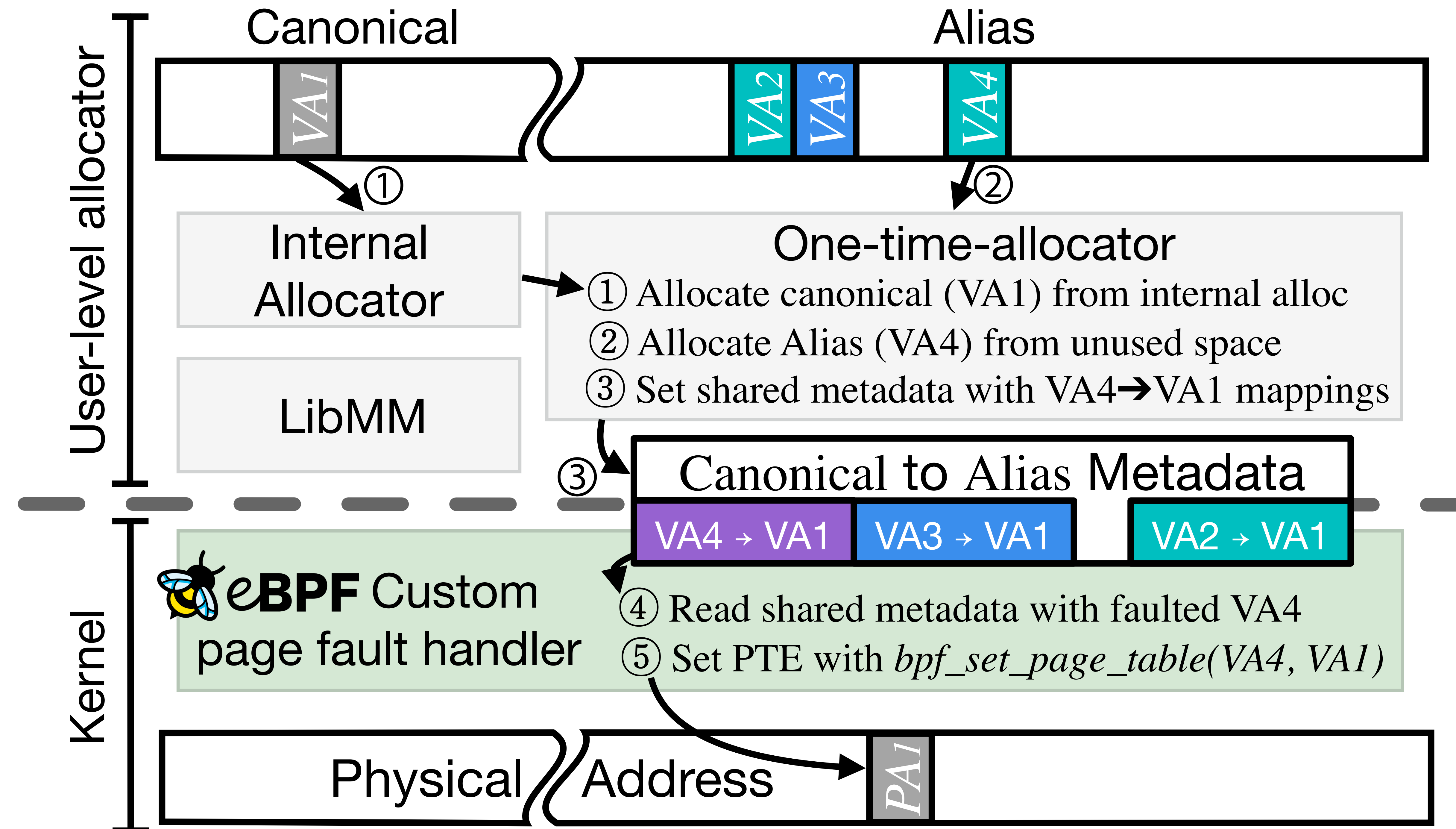
- BUDAlloc shows the **highest scalability** despite the **superior precision of detection**

High performance, Low memory overhead, Good scalability, and Superior precision of bug detection

KAIST

CASYS
Computer Architecture and Systems Lab

Summary



- We **decouple** virtual address from the kernel using **eBPF**
- By decoupling, we remove **semantic gap**, and introduce several **optimizations**
- BUDAlloc shows the **superior** detectability and performance

KAIST

CASYS
Computer Architecture and Systems Lab

Questions

- GITHUB: https://github.com/casys-kaist/BUDAlloc
- EMAIL: junhoahn@kaist.ac.kr
- CV: https://sites.google.com/view/junhoahn/

KAIST

CASYS
Computer Architecture and Systems Lab