Moneta: Ex-Vivo GPU Driver Fuzzing by Recalling In-Vivo Execution States

Joonkyo Jung*, Jisoo Jang*, Yongwan Jo, Jonas Vinck,
Alexios Voulimeneas, Stijn Volckaert, Dokyung Song

* Equal contribution.

YONSEI UNIVERSITY

KU LEUVEN

TU Delft

GPU Drivers are Vulnerable

Security Bulletin: NVIDIA GPU Display Driver - October 2024

27 High Severity CVEs

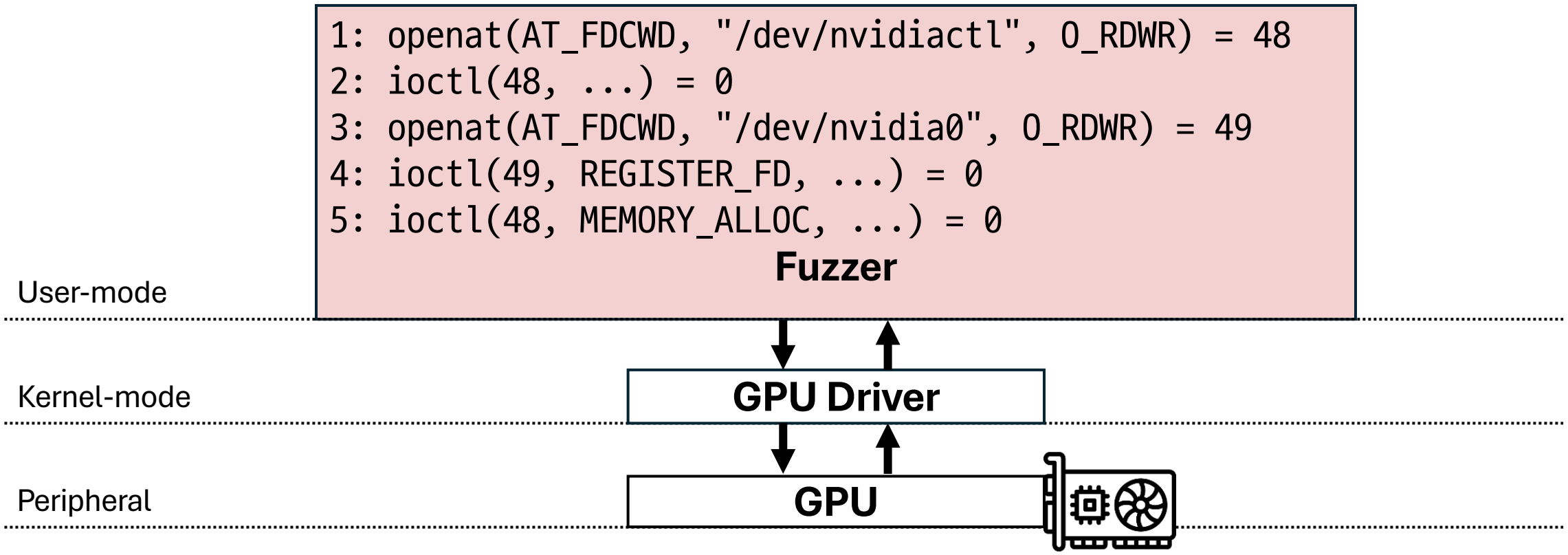
DETAILS

This section summarizes the potential vulnerabilities that this security update addresses and their impact. Descriptions use CWE™, and base scores and vectors use CVSS v3.1 standards.

NVIDIA GPU DISPLAY DRIVER

CVE ID	Description	Vector	Base Score	Severity	CWE	Impacts
CVE-2024-0126	NVIDIA GPU Display Driver for Windows and Linux contains a vulnerability which could allow a privileged attacker to escalate permissions. A successful exploit of this vulnerability might lead to code execution, denial of service, escalation of privileges, information disclosure, and data tampering.	AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H	8.2	High	CWE-20	Code execution, denial of service, escalation of privileges, information disclosure, and data tampering
	NVIDIA GPU Display Driver for Windows					

Fuzzing Device Drivers



[1] Pailoor et al., "MoonShine: Optimizing OS fuzzer seed selection with trace distillation." *USENIX Security 18*.

Inter-Syscall Dependency Challenge

Value Dependency

```
1: openat(AT_FDCWD, "/dev/nvidiactl", O_RDWR) = 48
2: ioctl(48, ...) = 0
3: openat(AT_FDCWD, "/dev/nvidia0", O_RDWR) = 49
4: ioctl(49, REGISTER_FD, ...) = 0
5: ioctl(48, MEMORY_ALLOC, ...) = 0
```

Fuzzer

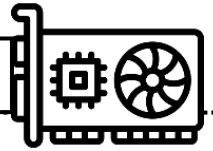
User-mode

Kernel-mode

GPU Driver

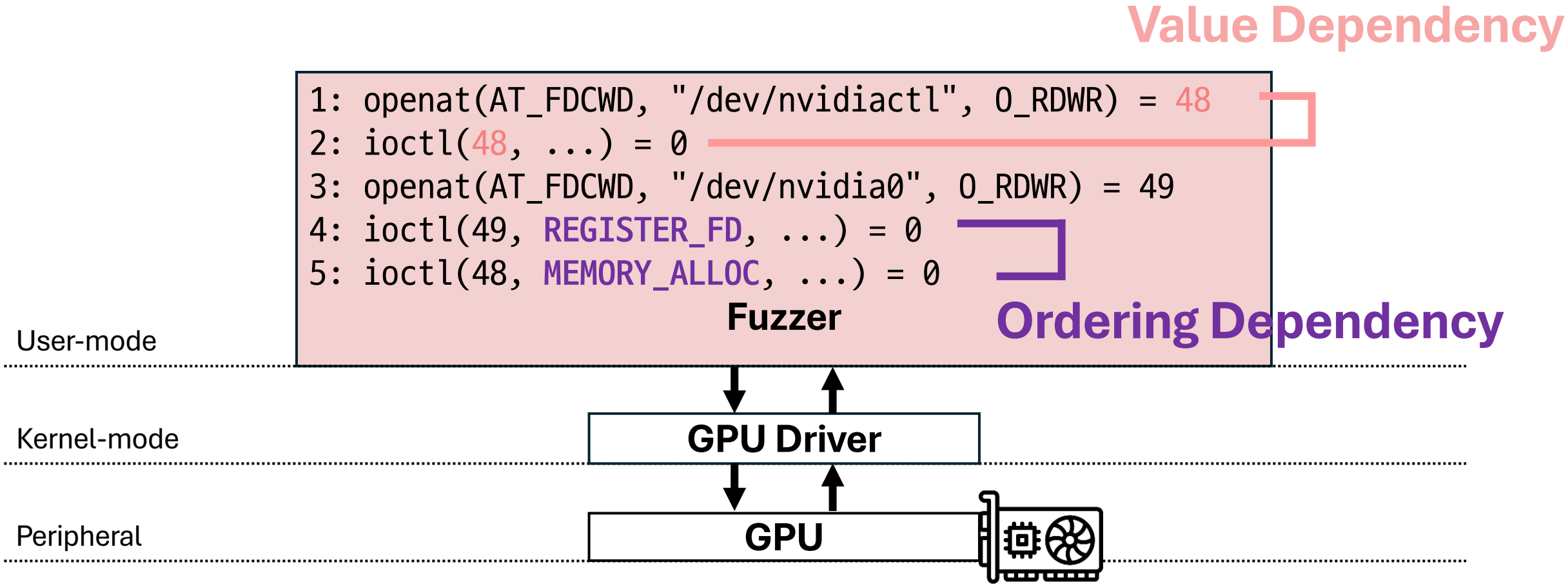
Peripheral

GPU



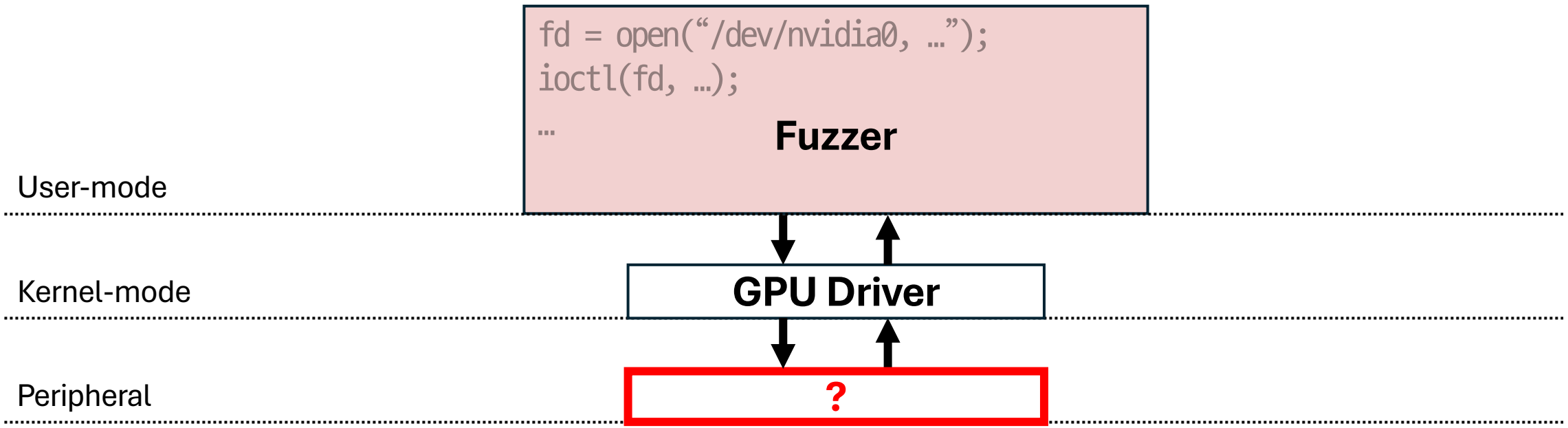
[1] Pailoor et al., "MoonShine: Optimizing OS fuzzer seed selection with trace distillation." *USENIX Security 18*.

Inter-Syscall Dependency Challenge

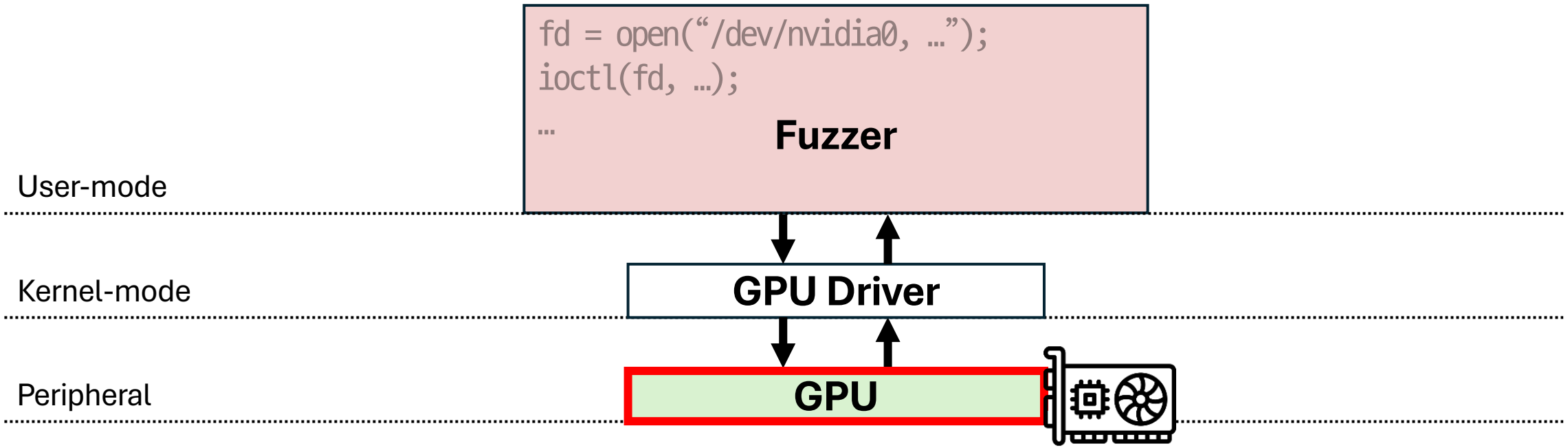


[1] Pailoor et al., "MoonShine: Optimizing OS fuzzer seed selection with trace distillation." *USENIX Security 18*.

Hardware Dependency Challenge



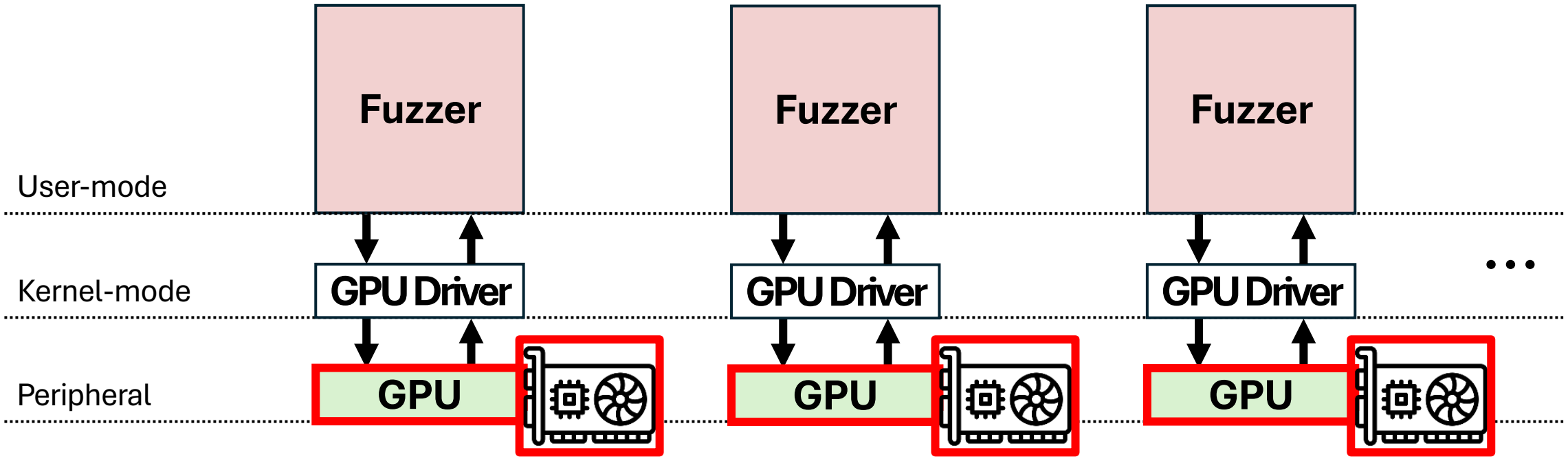
Addressing Hardware Dependency via In-vivo Approaches[2]



[2] Talebi et al., "Charm: Facilitating dynamic analysis of device drivers of mobile systems." *USENIX Security '18*.

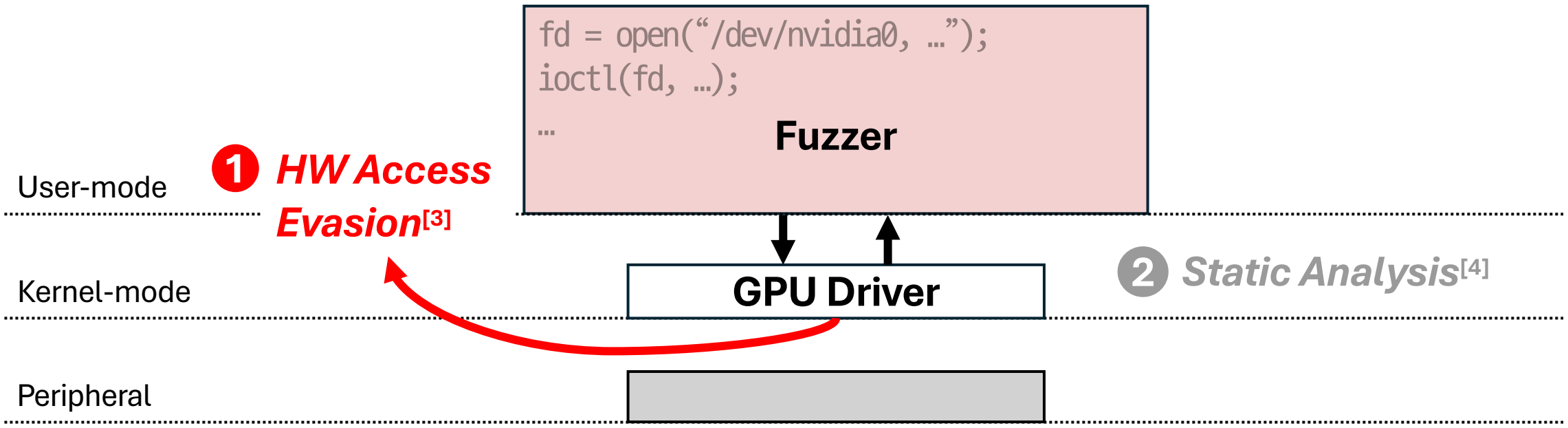
Downside of In-vivo Approaches[2]

Hard to scale!



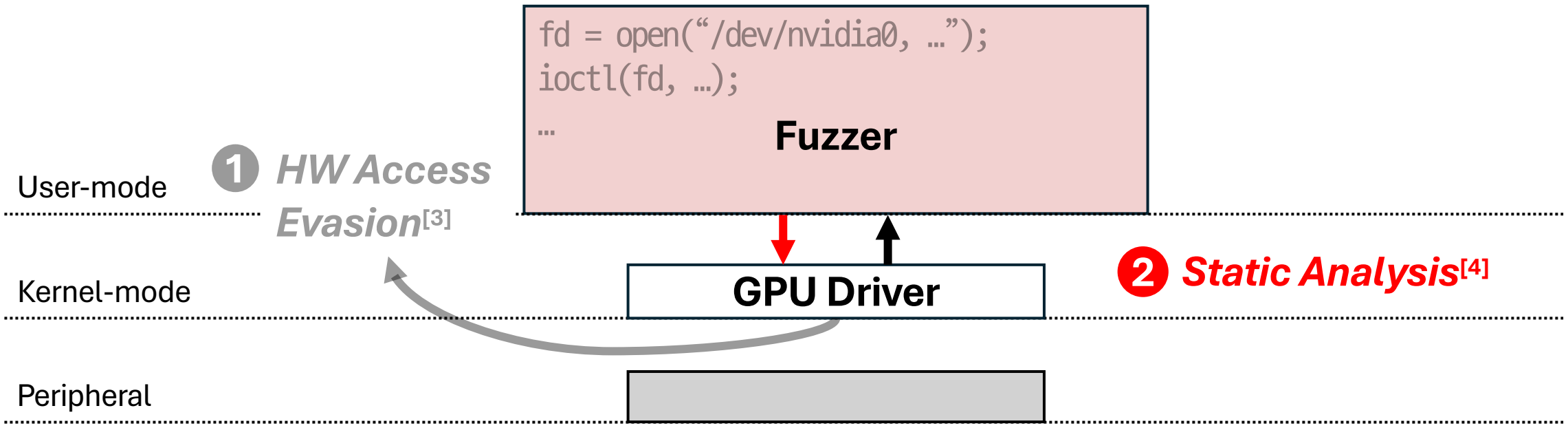
[2] Talebi et al., "Charm: Facilitating dynamic analysis of device drivers of mobile systems." *USENIX Security '18*.

Addressing Hardware Dependency via Ex-vivo Approaches



[3] Pustogarov et al., "Ex-vivo dynamic analysis framework for Android device drivers." *IEEE S&P '20*.
[4] Shen et al., "Drifuzz: Harvesting Bugs in Device Drivers from Golden Seeds." *USENIX Security '22*.

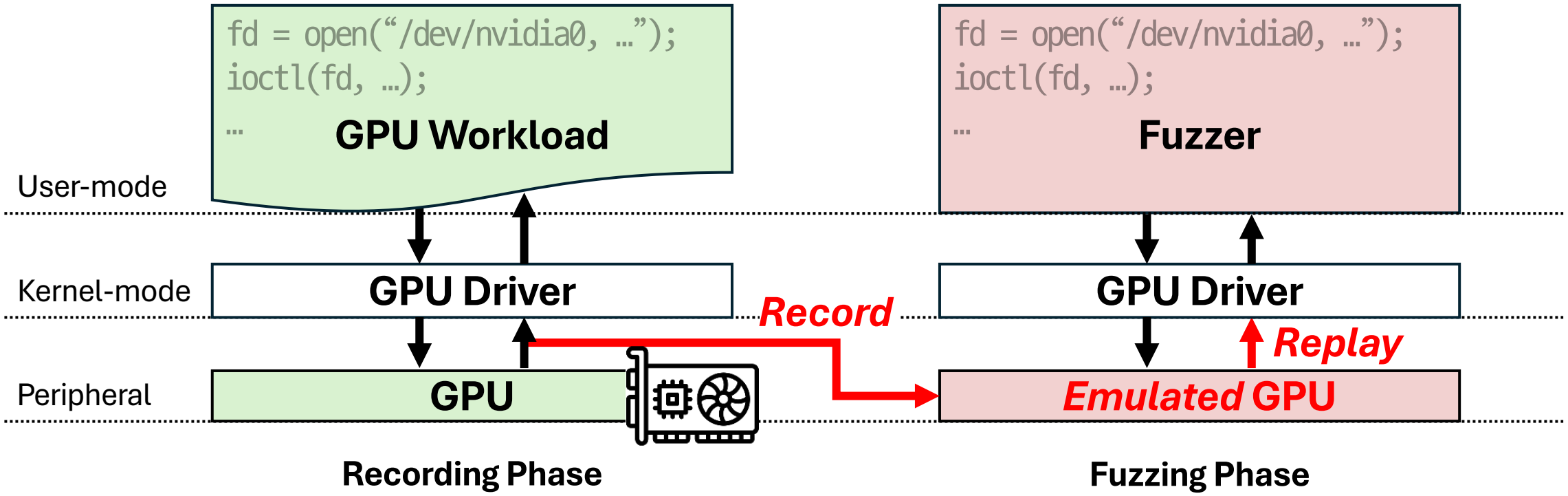
Addressing Hardware Dependency via Ex-vivo Approaches



[3] Pustogarov et al., "Ex-vivo dynamic analysis framework for Android device drivers." *IEEE S&P '20*.
[4] Shen et al., "Drifuzz: Harvesting Bugs in Device Drivers from Golden Seeds." *USENIX Security '22*.

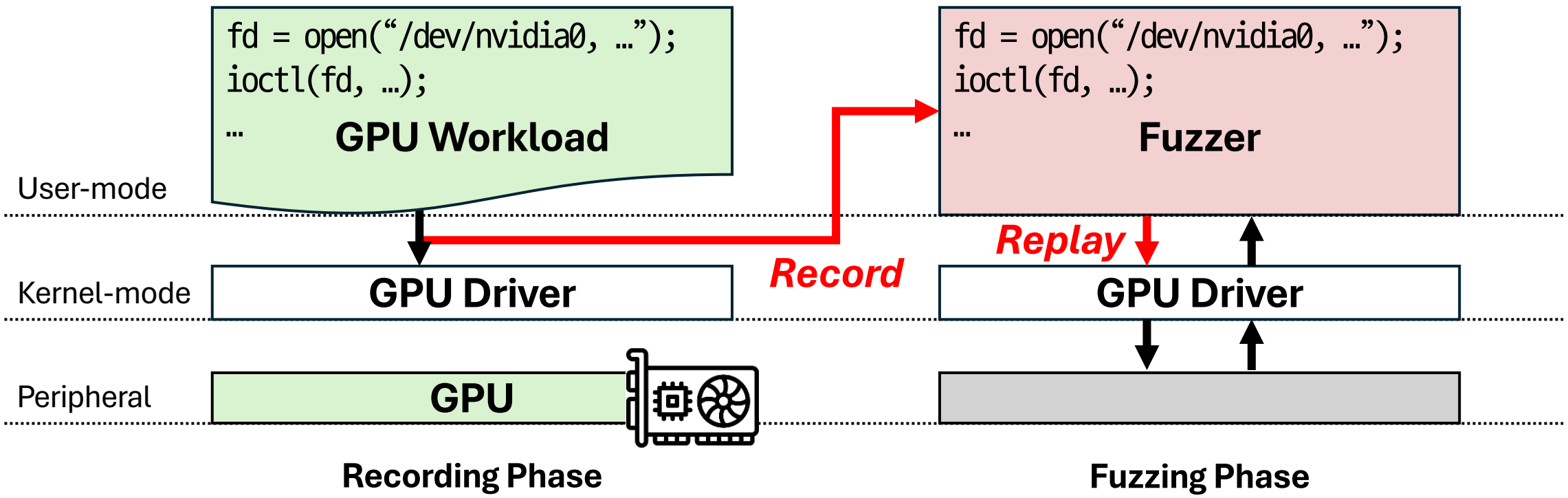
Addressing Hardware Dependency via Ex-vivo Approaches

❸ *Record-and-Replay*[5]



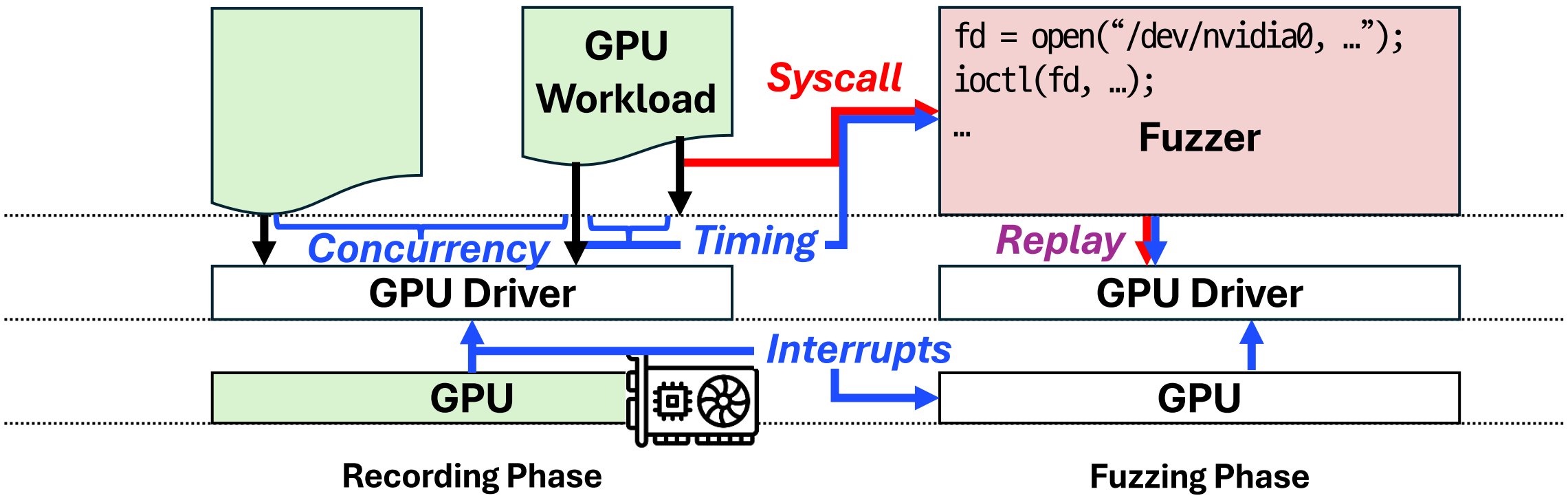
[5] Maier et al., "Bsod: Binary-only scalable fuzzing of device drivers." *RAID '21*.

Addressing Inter-Syscall Dependency via Record-and-Replay



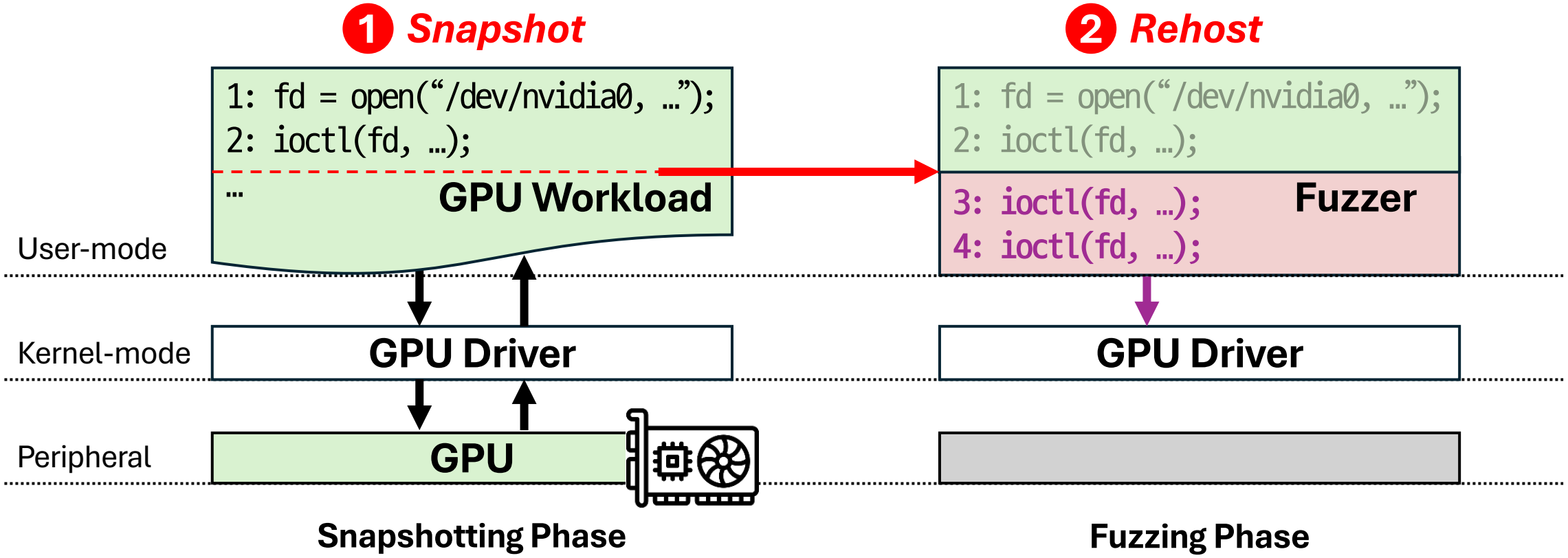
[1] Pailoor et al., "MoonShine: Optimizing OS fuzzer seed selection with trace distillation." *USENIX Security 18*.

Nondeterminism Challenge in Record-and-Replay

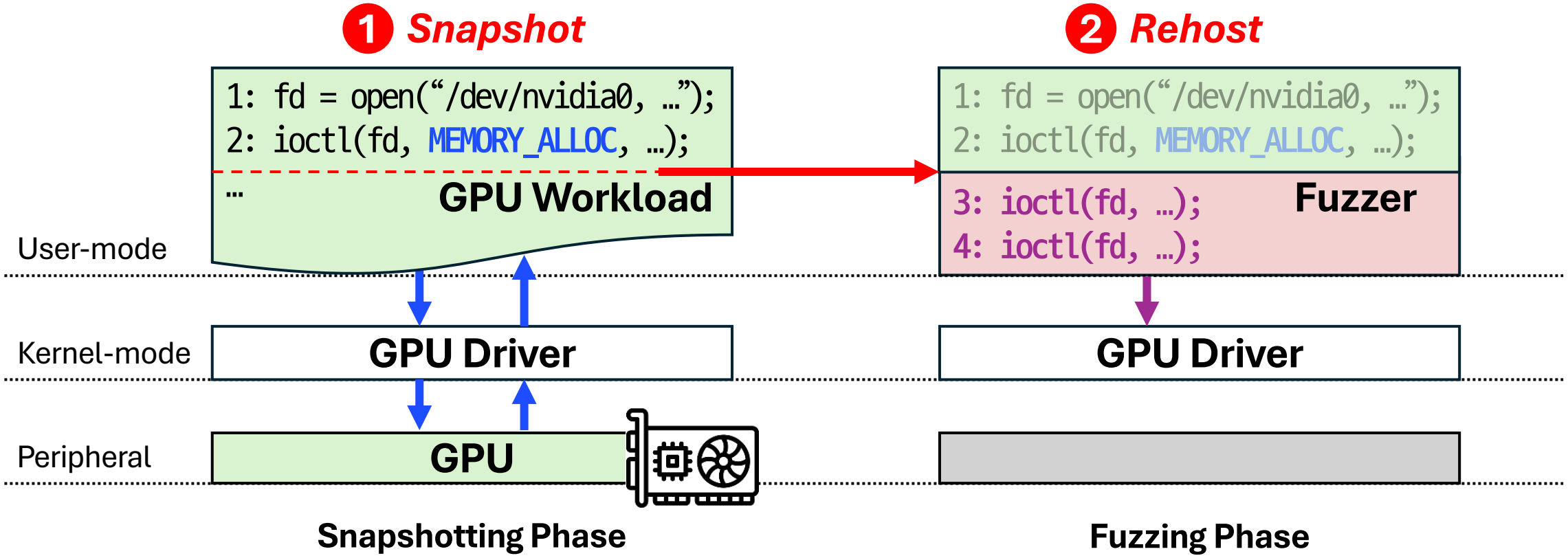


[6] Jang et al., "ReUSB: Replay-Guided USB Driver Fuzzing." *USENIX Security '23*.

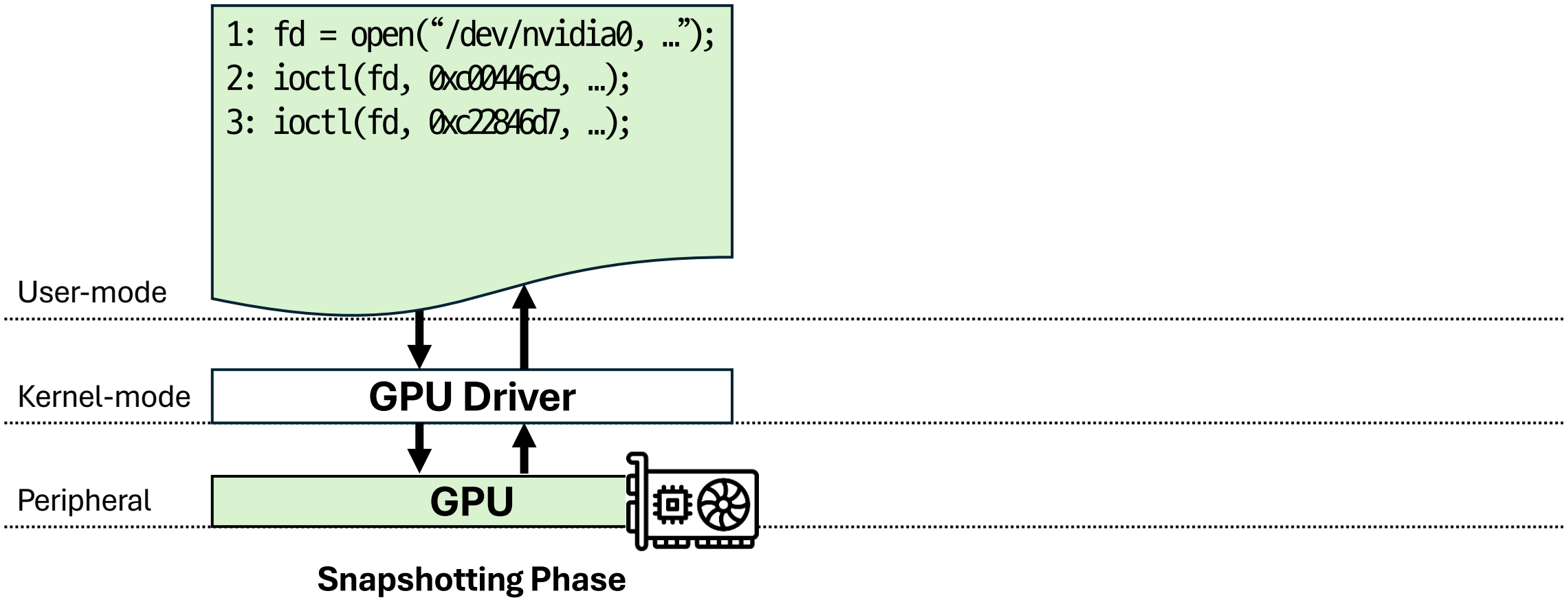
Addressing Nondeterminism via Snapshot-and-Rehost



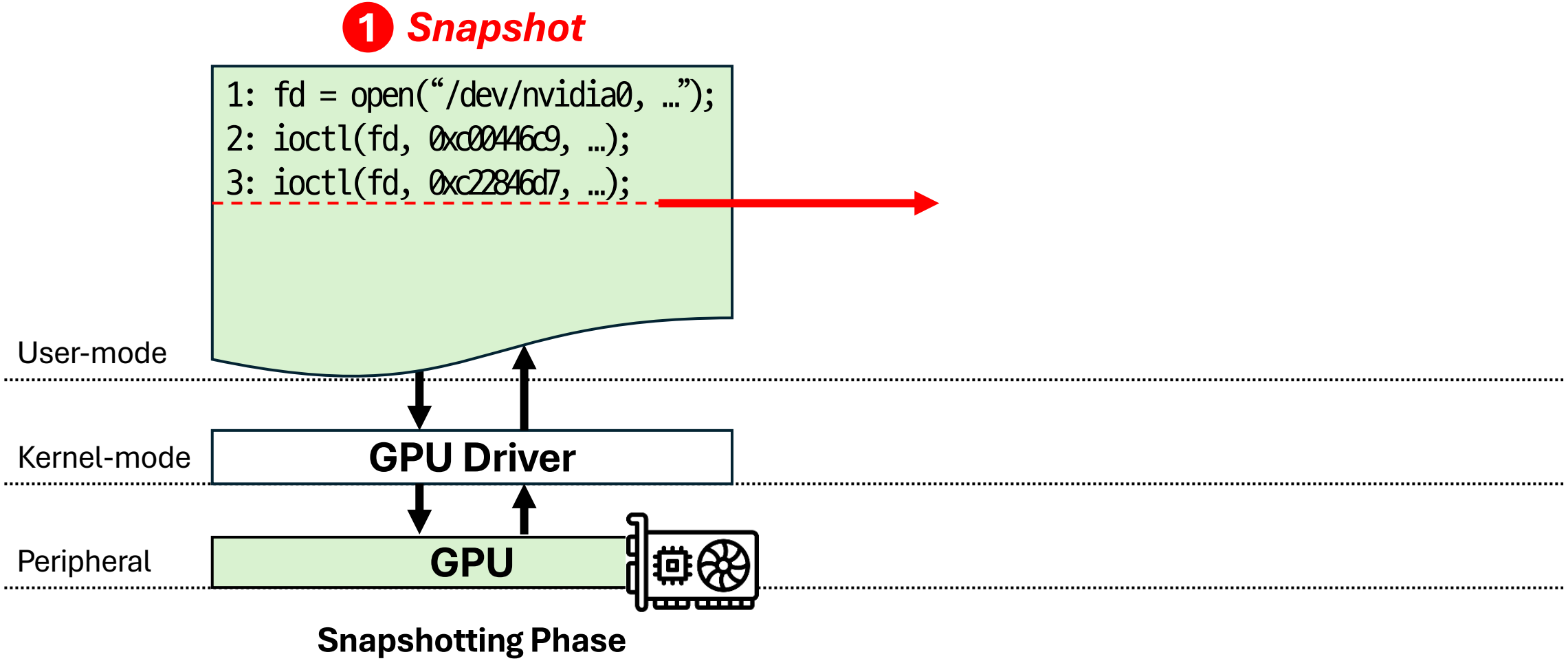
Addressing Hardware Dependency via Snapshot-and-Rehost



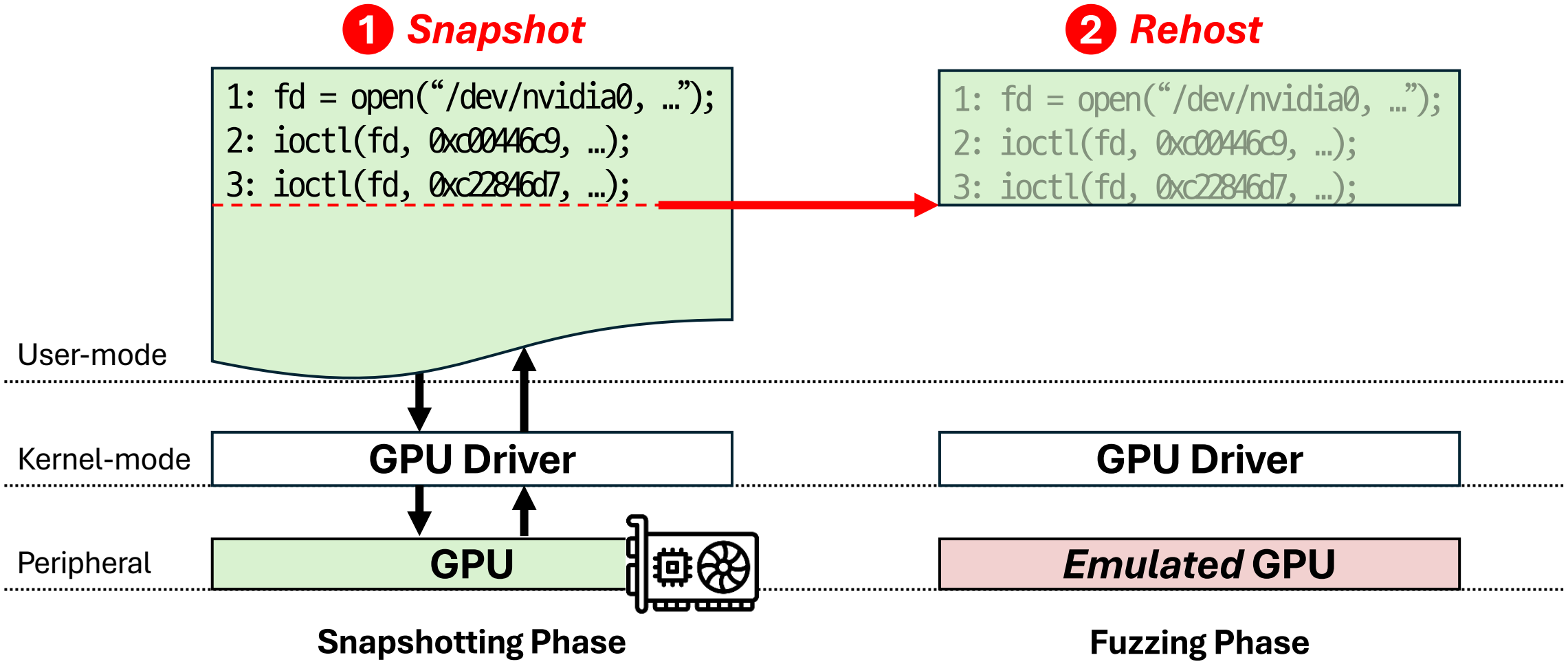
Moneta: Snapshot-and-Rehost + Record-and-Replay



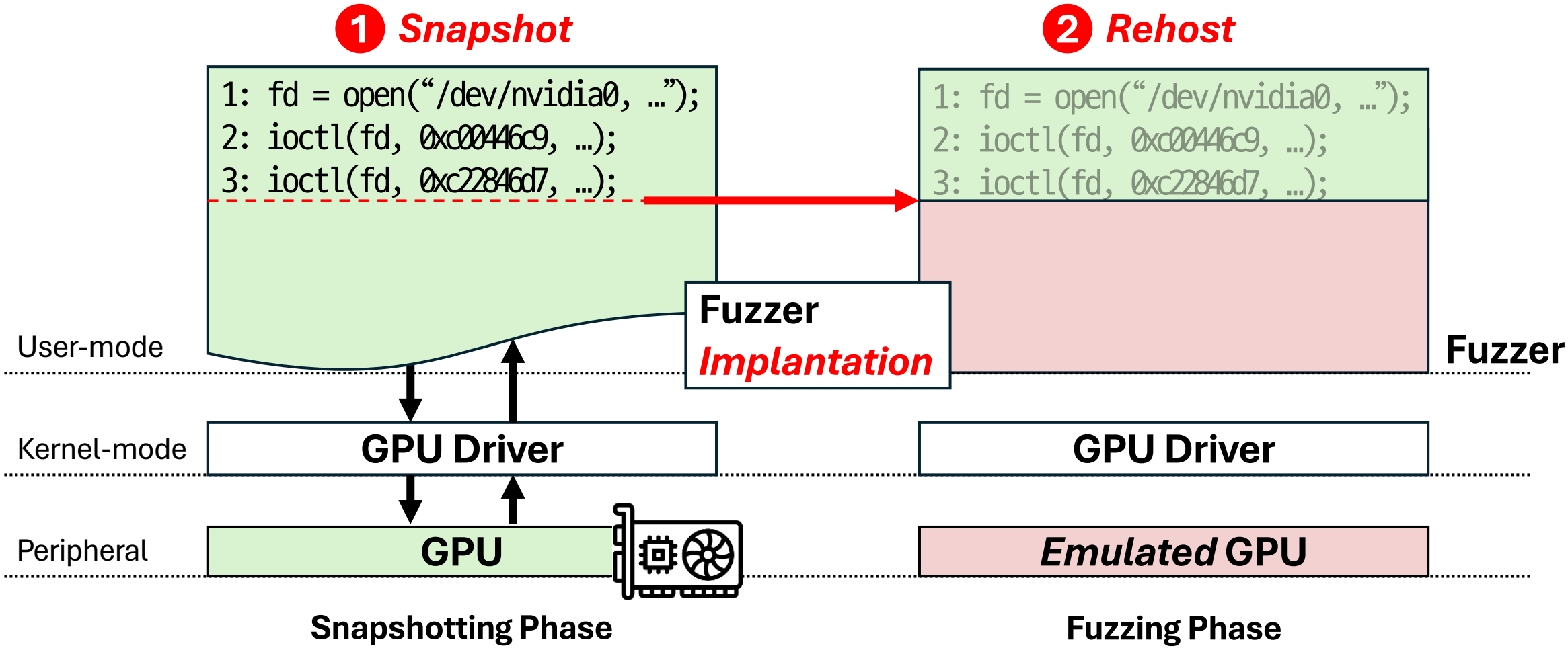
Moneta: Snapshot-and-Rehost + Record-and-Replay



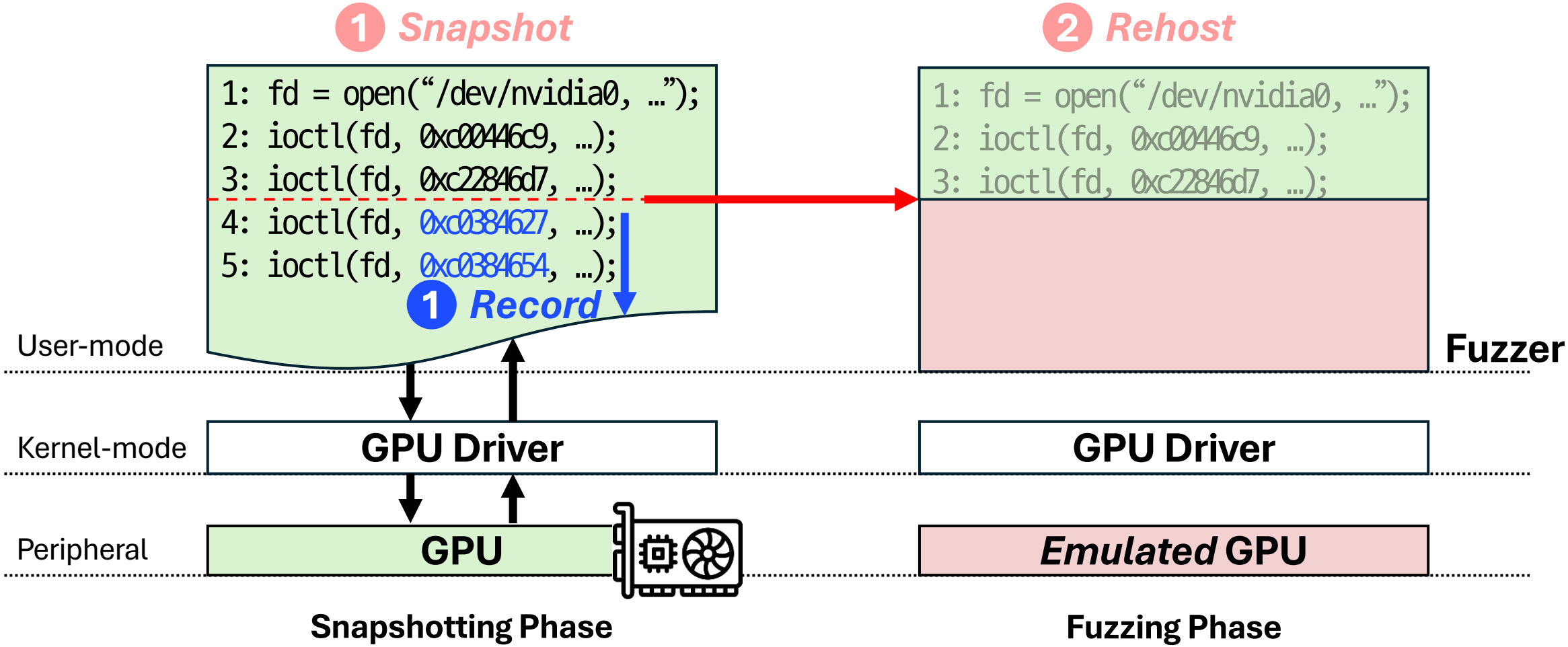
Moneta: Snapshot-and-Rehost + Record-and-Replay



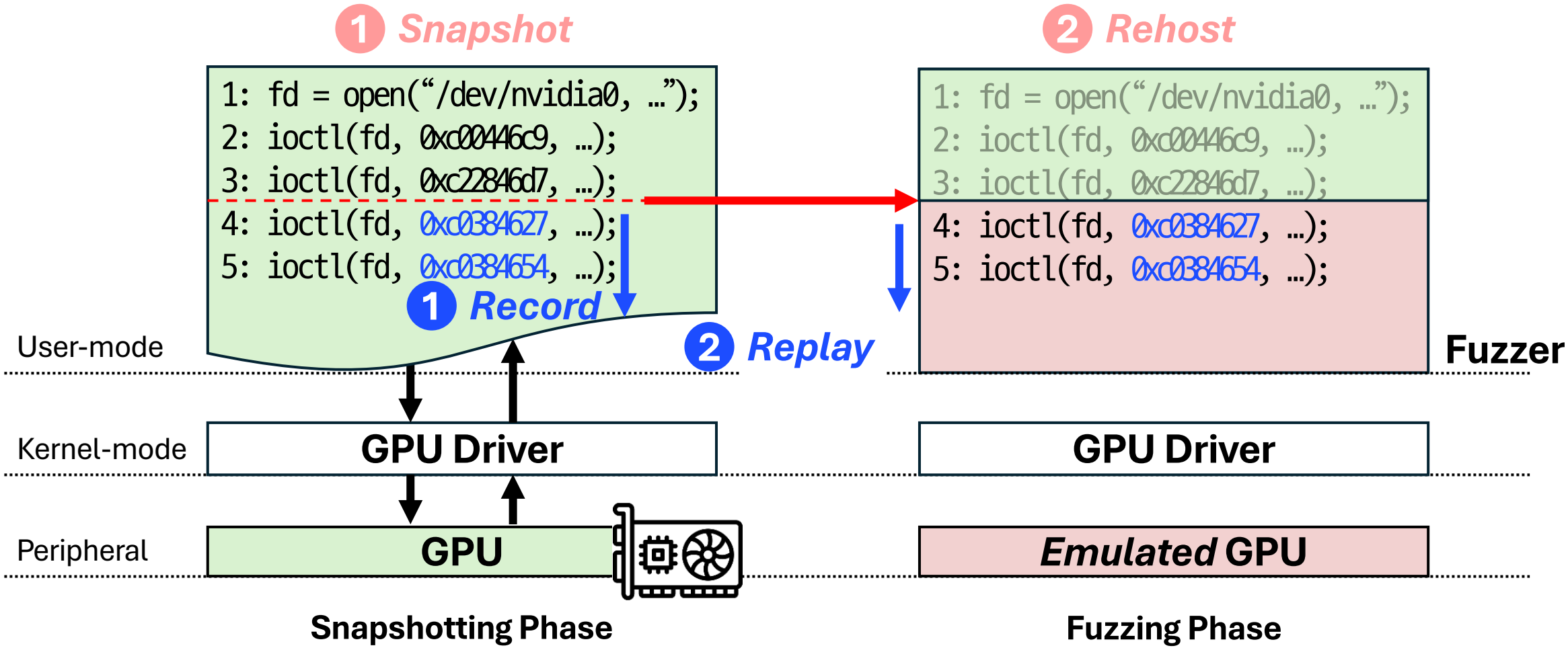
Moneta: Snapshot-and-Rehost + Record-and-Replay



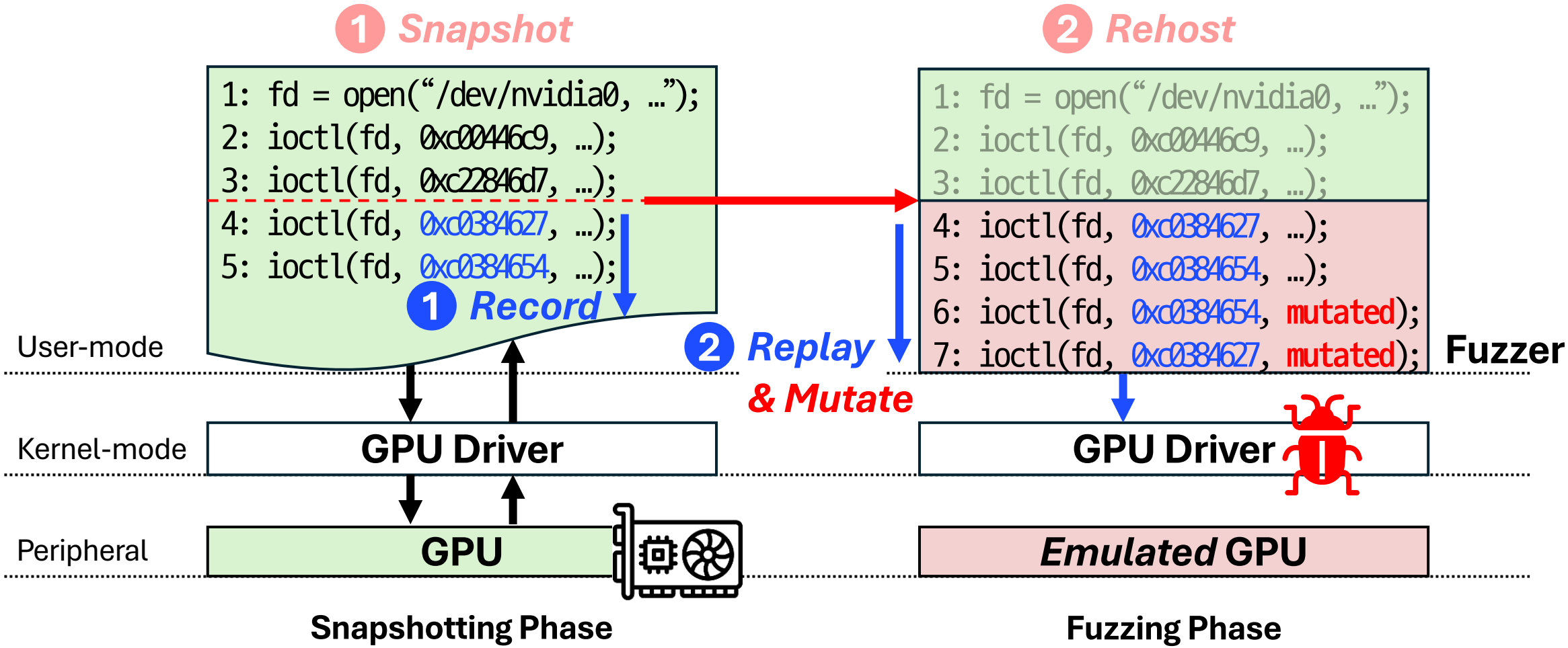
Moneta: Snapshot-and-Rehost + Record-and-Replay



Moneta: Snapshot-and-Rehost + Record-and-Replay



Moneta: Snapshot-and-Rehost + Record-and-Replay



Moneta: Snapshot-and-Rehost + Record-and-Replay

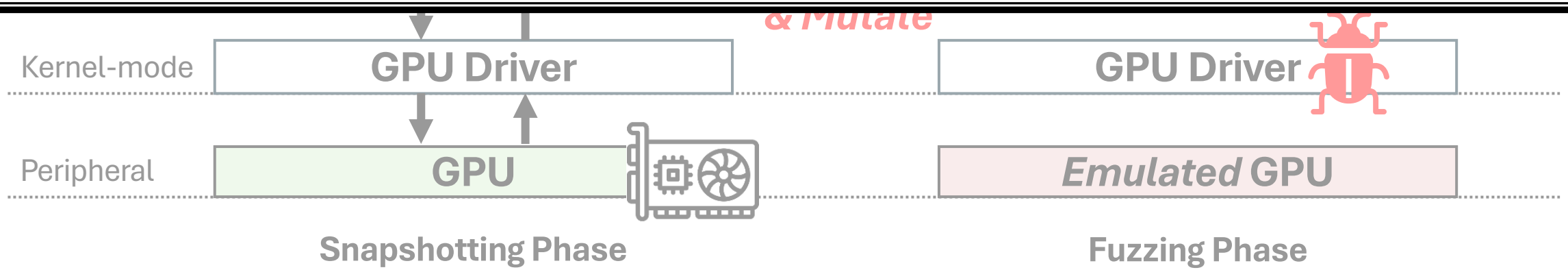
1 Snapshot

```
1: fd = open("/dev/nvidia0, …");
2: ioctl(fd, 0xc00446c9, …);
```

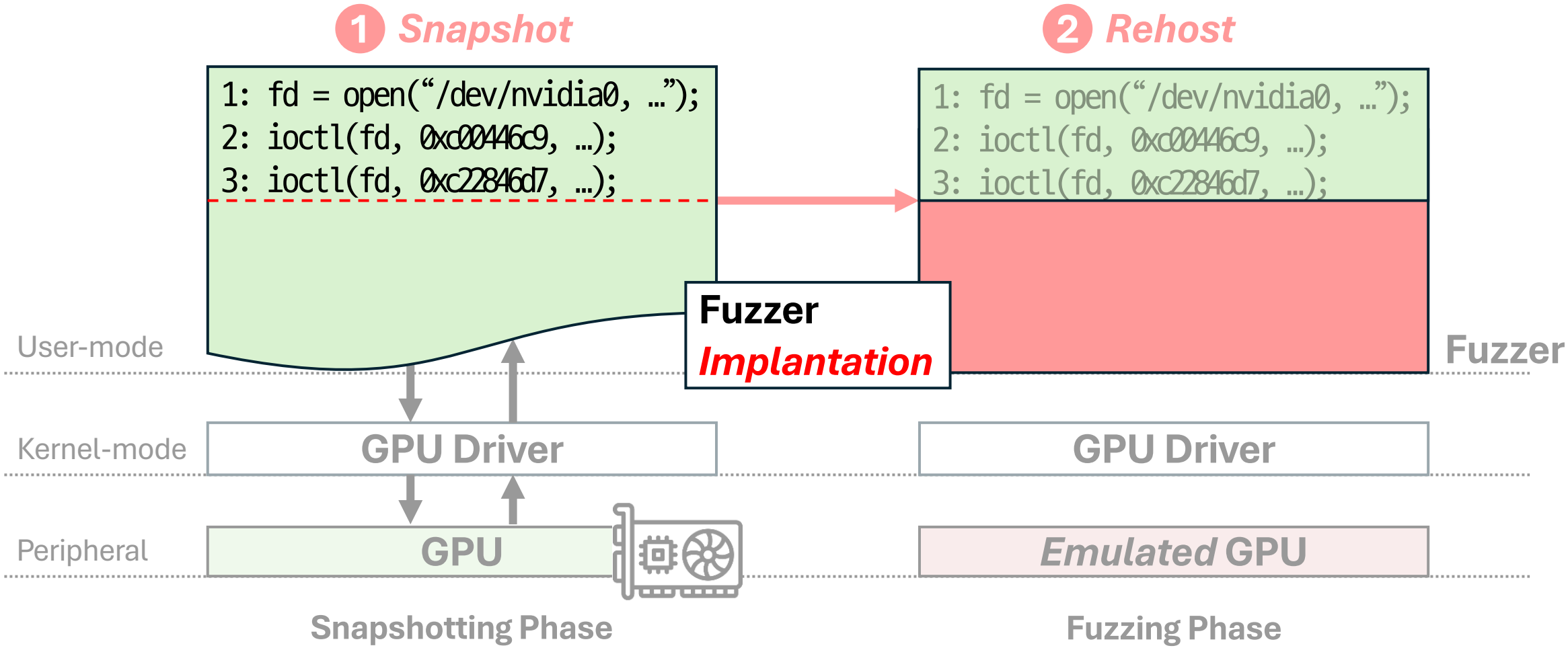
2 Rehost

```
1: fd = open("/dev/nvidia0, …");
2: ioctl(fd, 0xc00446c9, …);
```

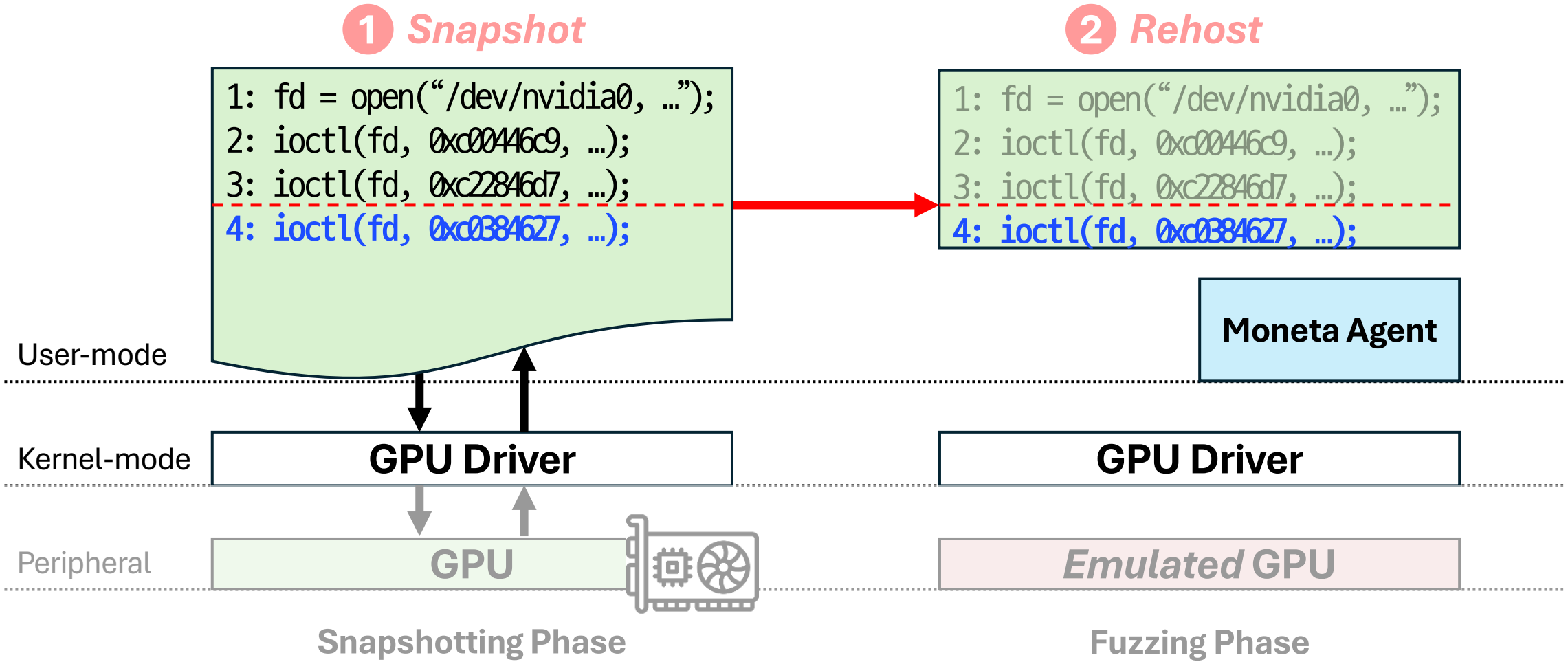
Combination of SnR and RnR **deterministically recalls** deep driver states while **still being capable** of evolutionary fuzzing.



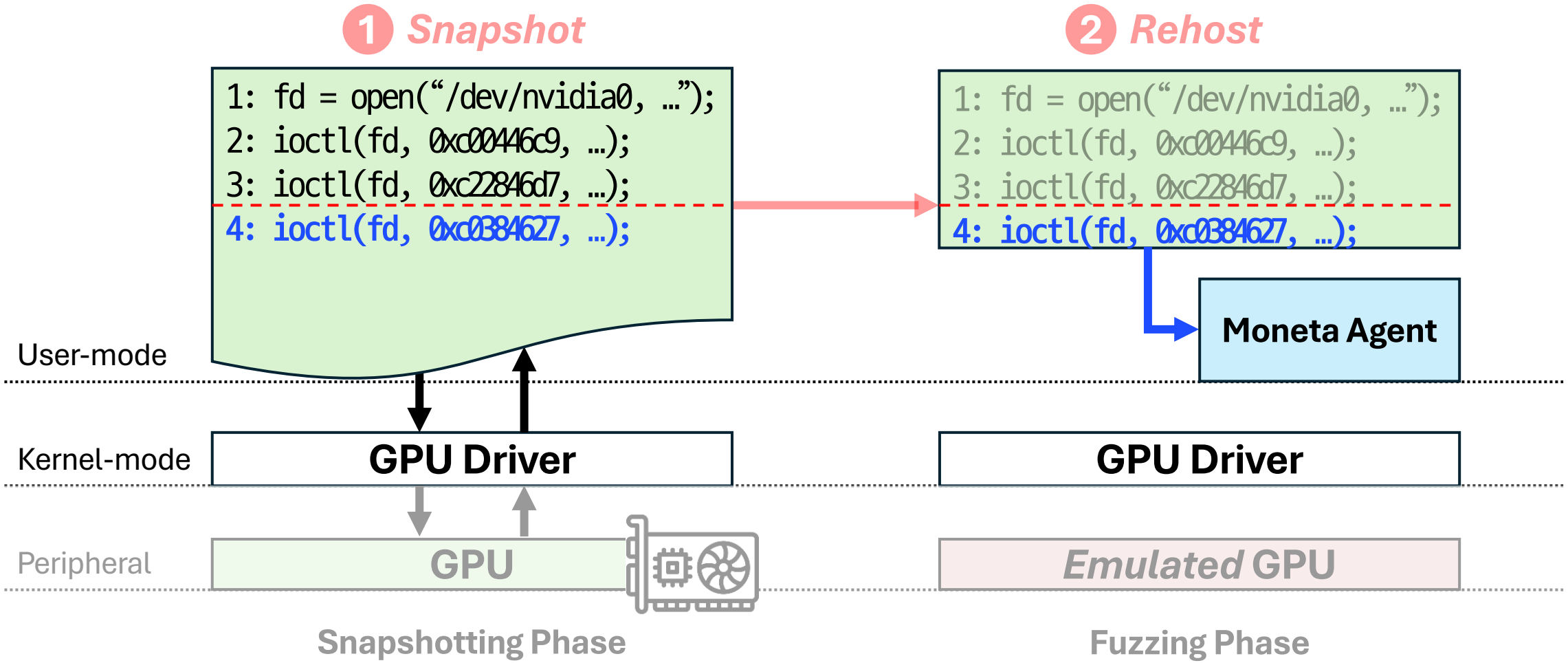
Fuzzer Implantation



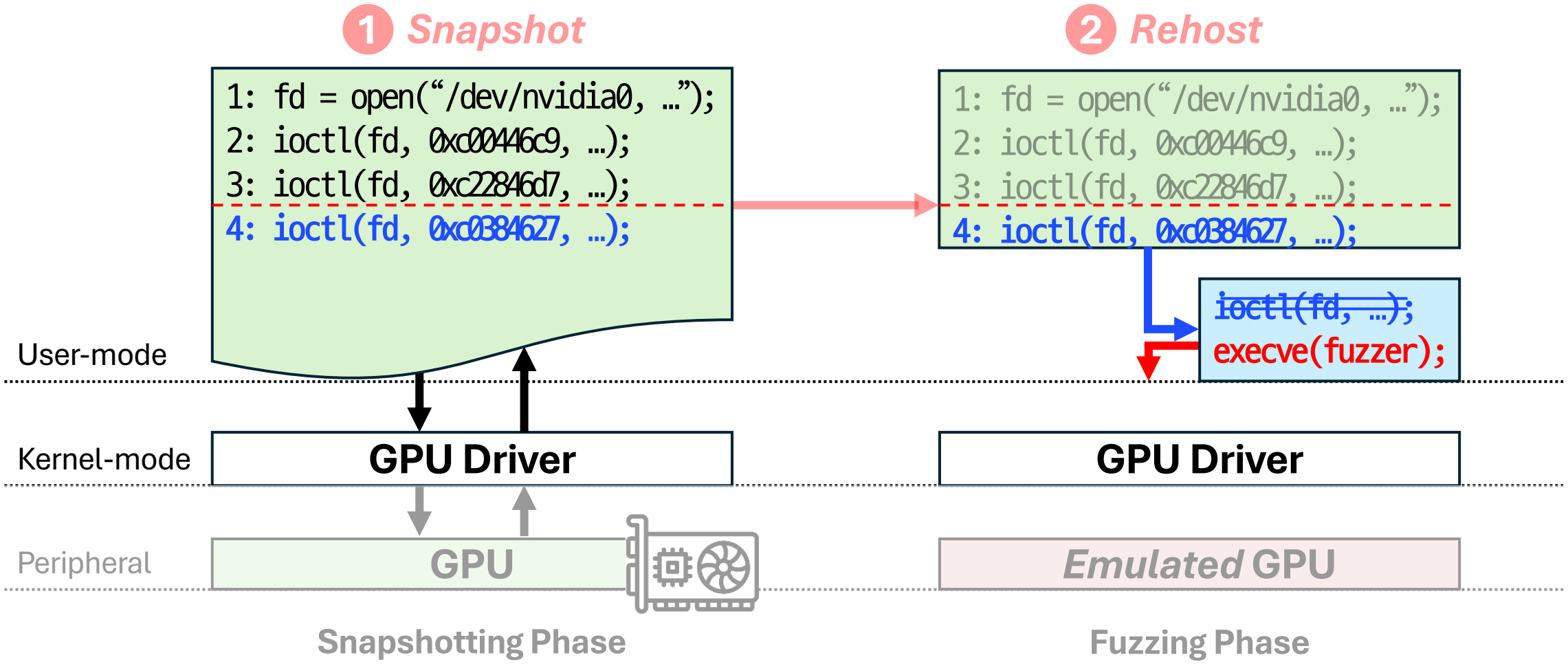
Fuzzer Implantation



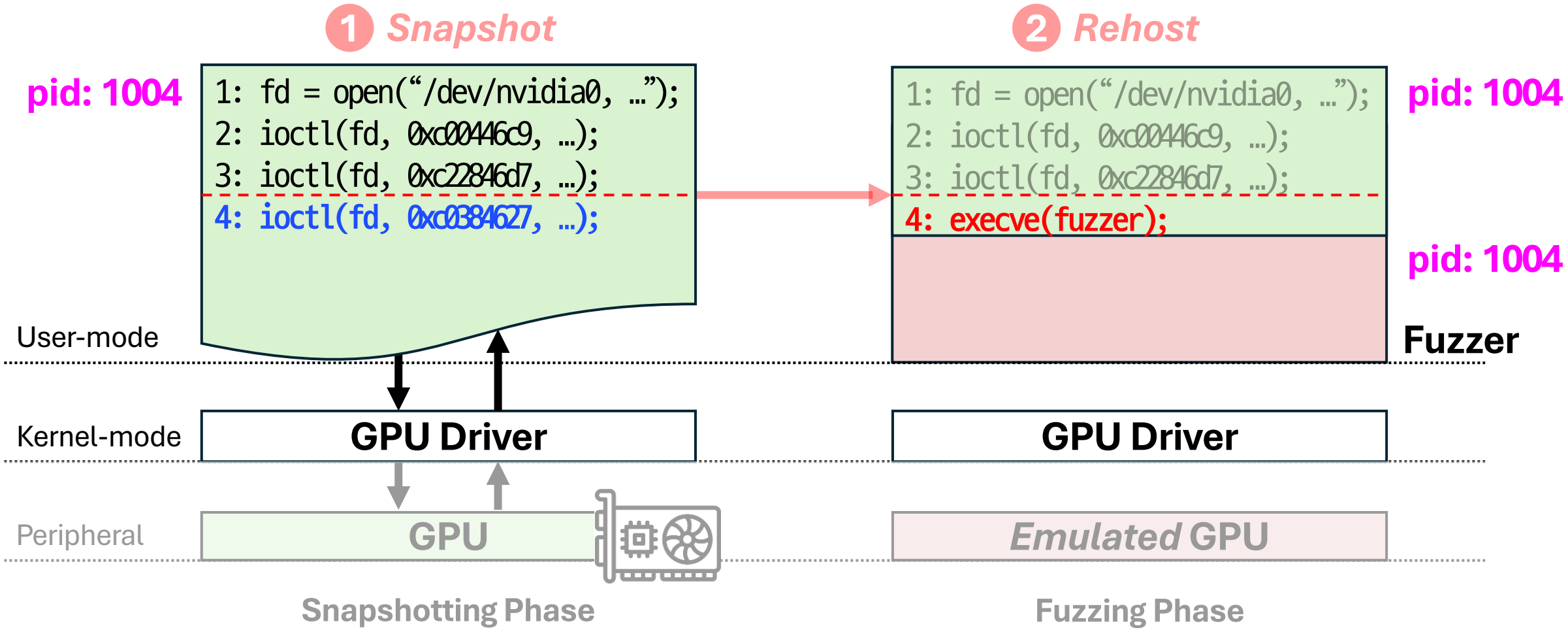
Fuzzer Implantation



Fuzzer Implantation

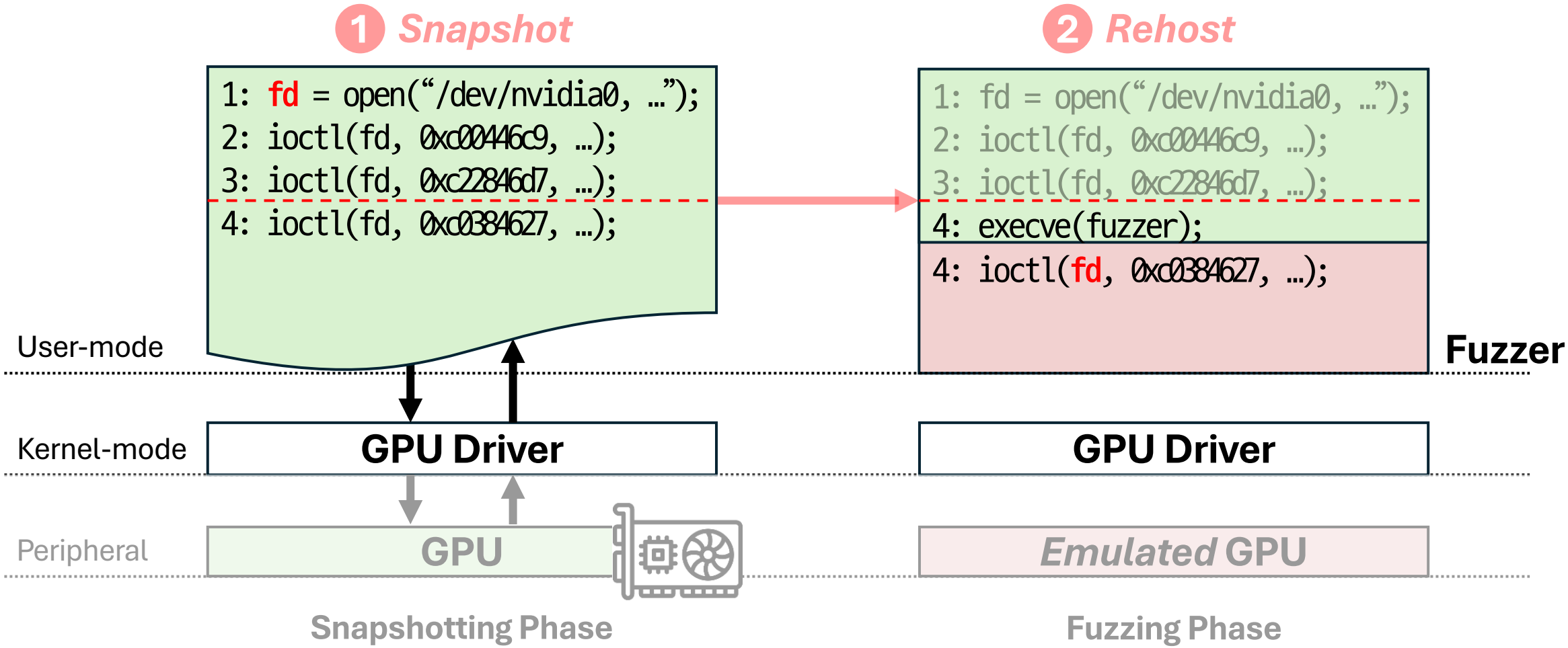


Fuzzer Implantation

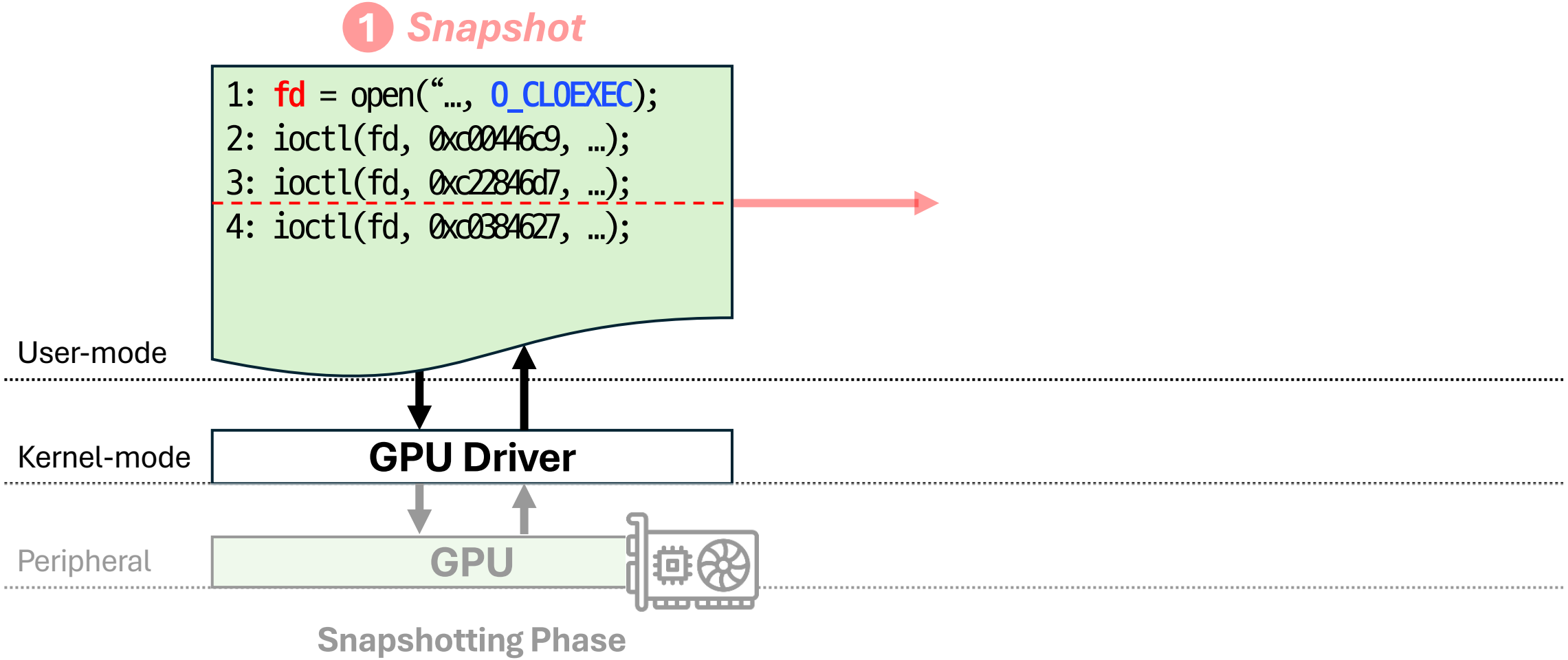


```
NV_STATUS rm_create_mmap_context(…)
{
…
    if ((rmStatus = rmapiLockAcquire(RMAPI_LOCK_FLAGS_READ, RM_LOCK_MODULES_OSAPI)) == NV_OK)
    {
        RmClient *pClient;
…
            if (pClient->ProcID != osGetCurrentProcess())
            {
                rmStatus = NV_ERR_INVALID_CLIENT;
            }
…
}
    return rmStatus;
}
```

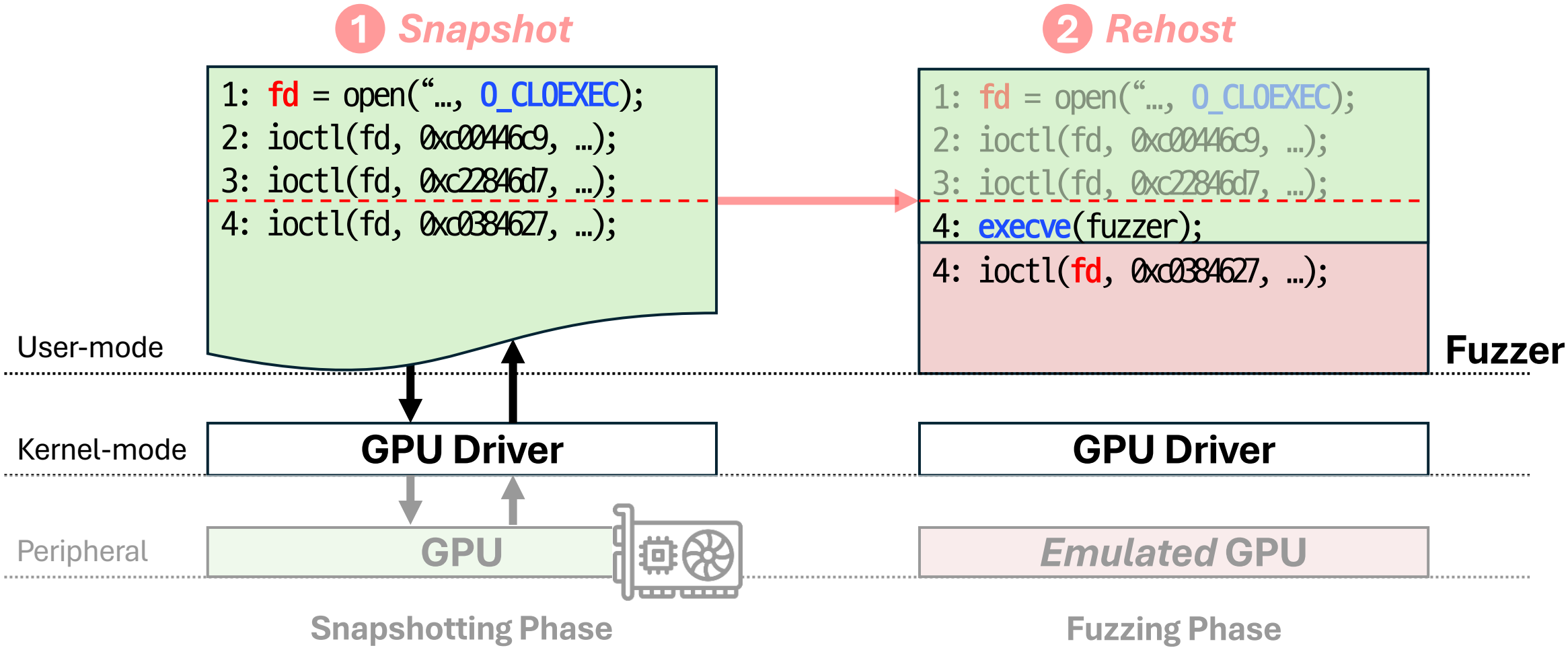
File Descriptor Preservation



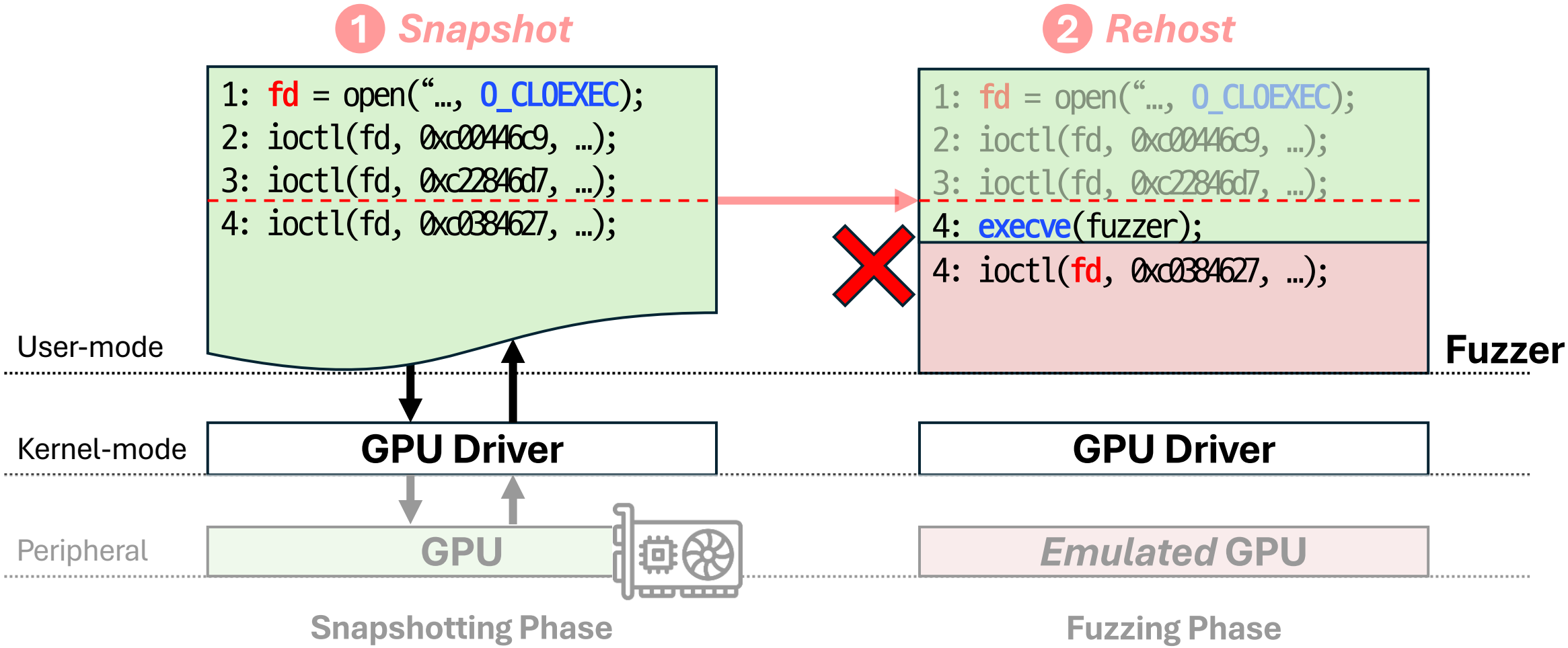
File Descriptor Preservation



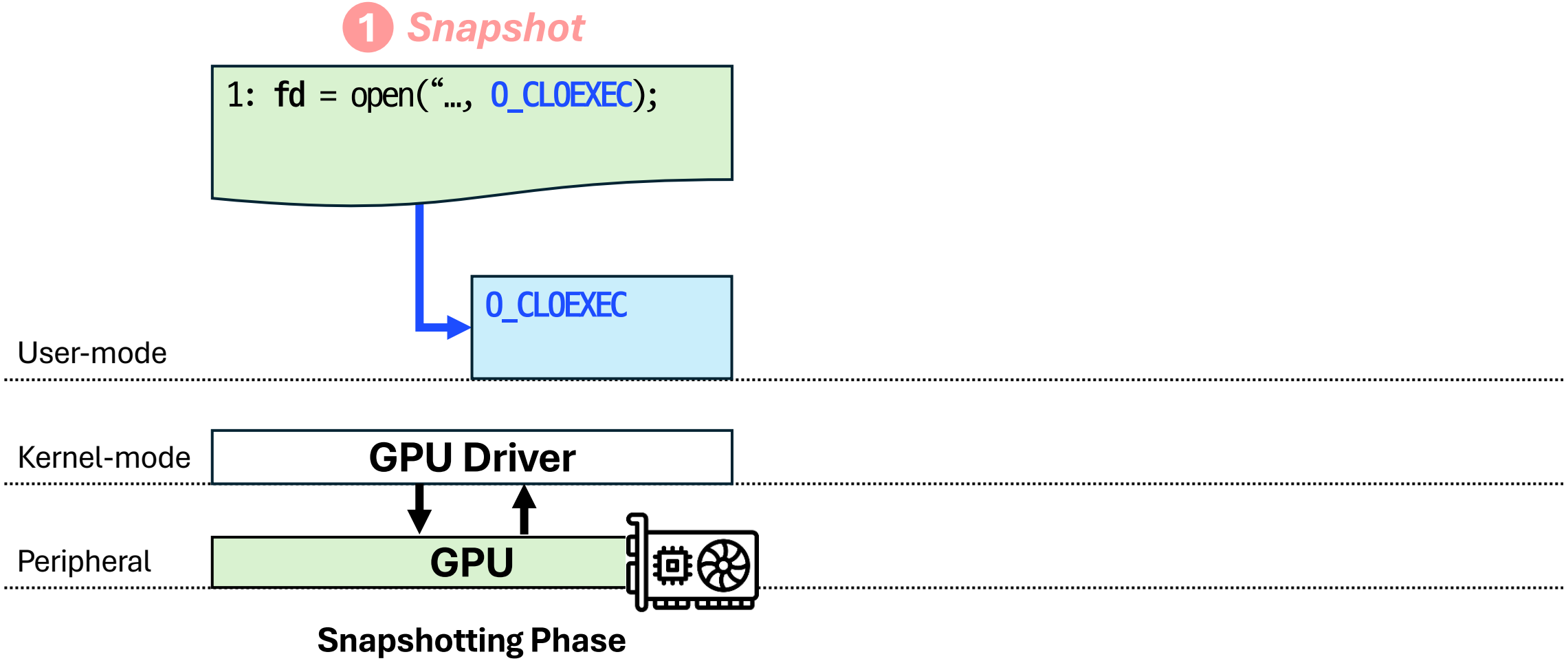
File Descriptor Preservation



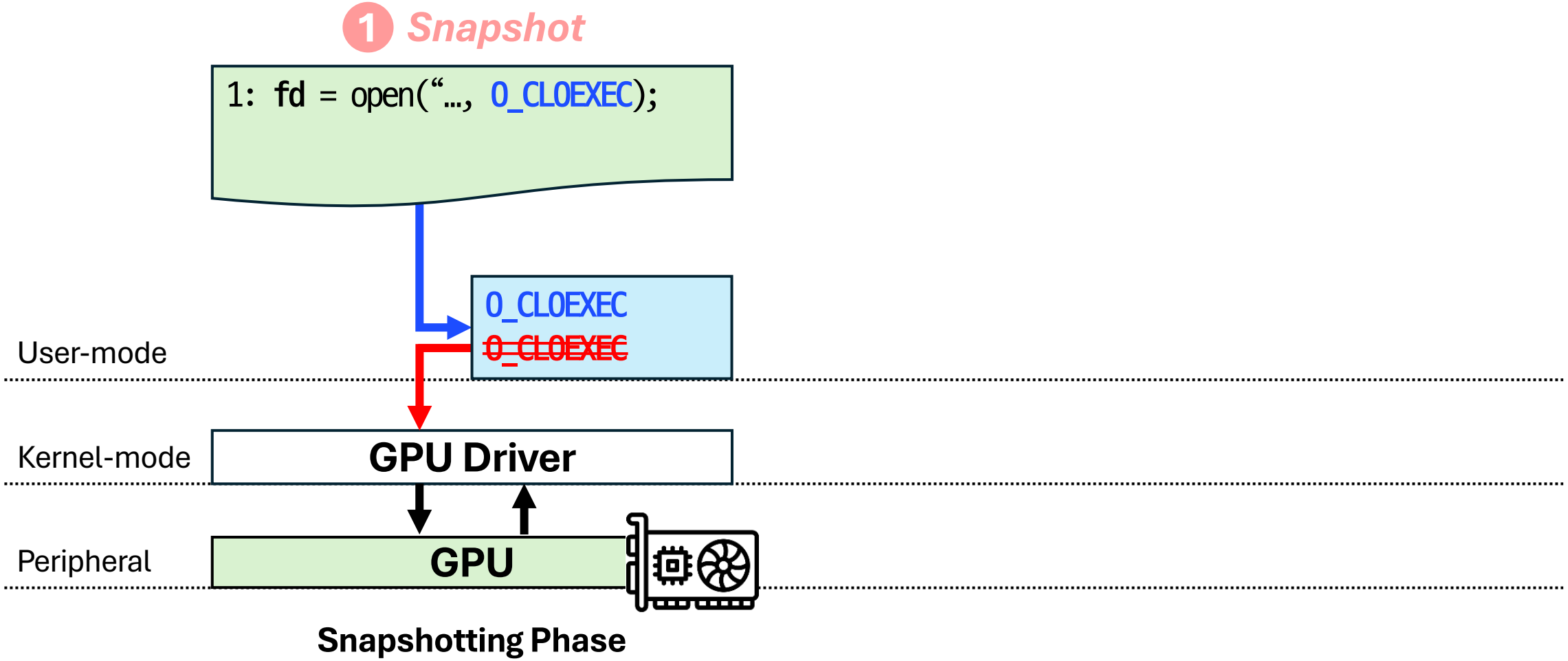
File Descriptor Preservation



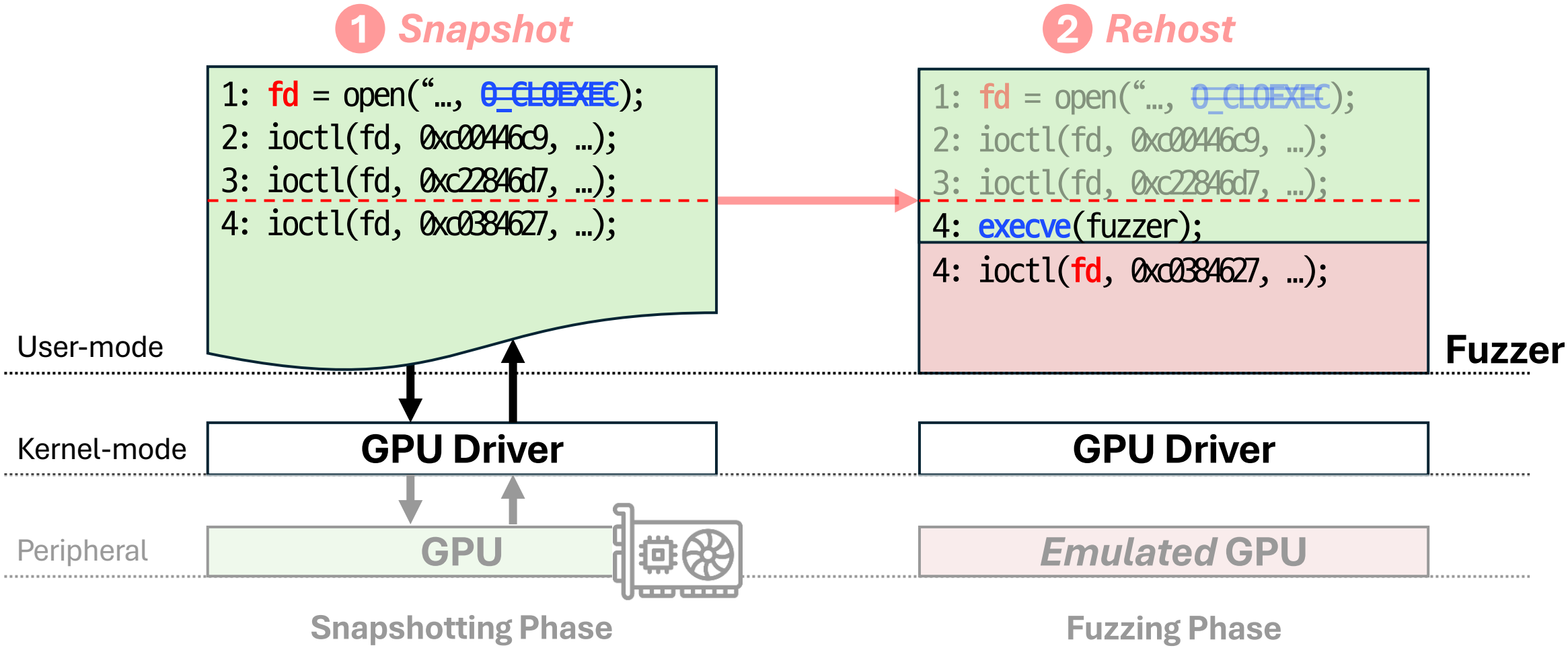
File Descriptor Preservation



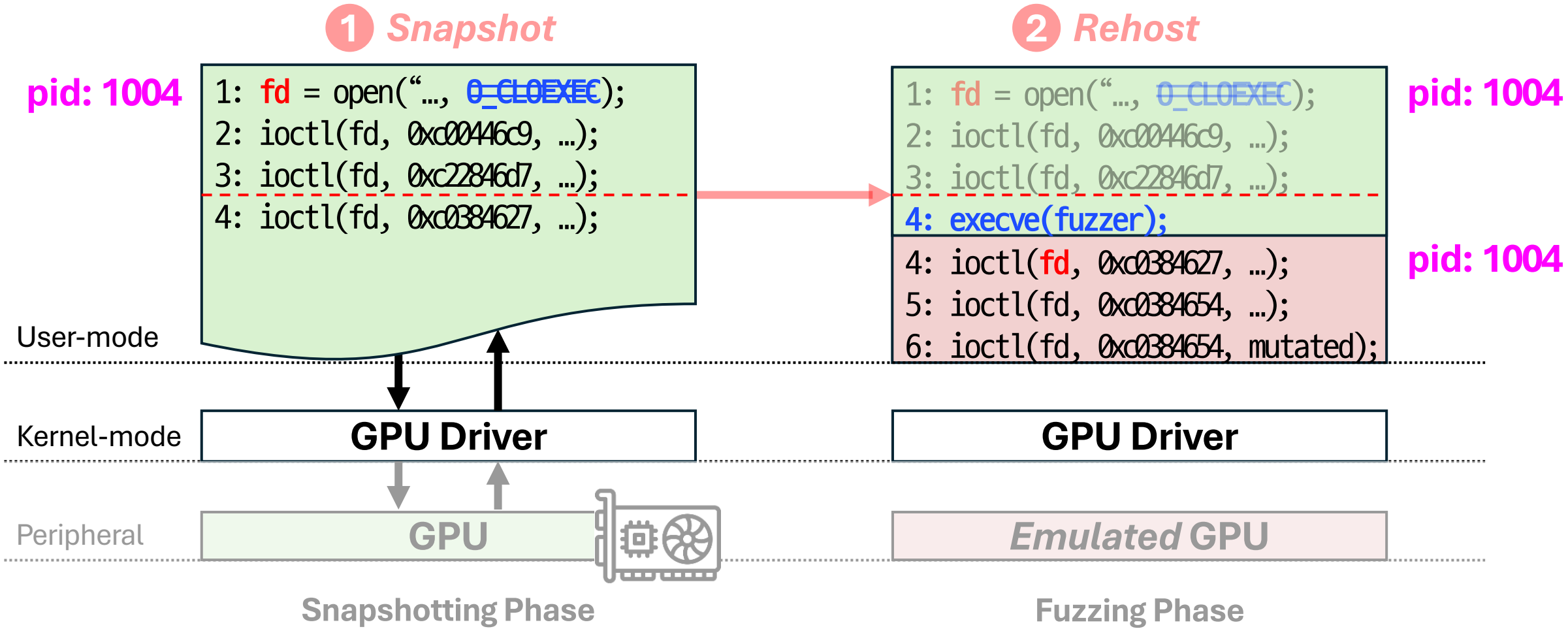
File Descriptor Preservation



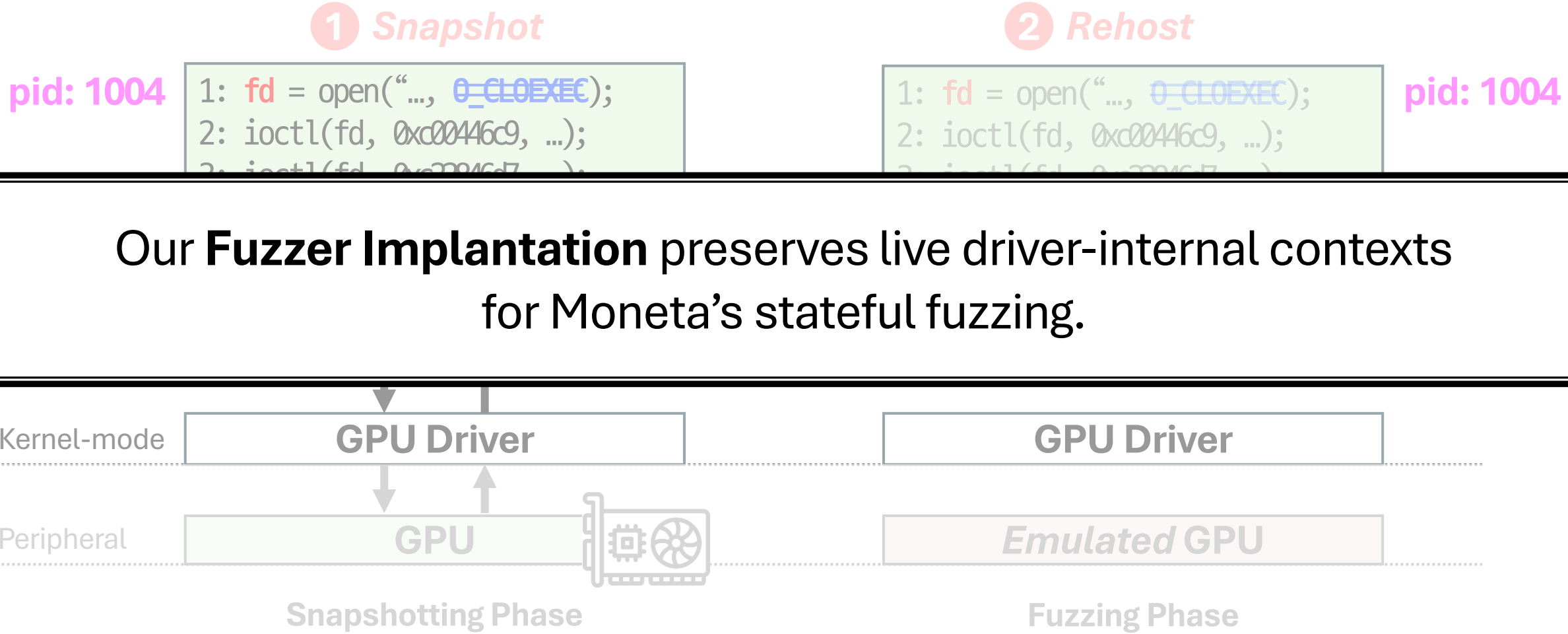
File Descriptor Preservation



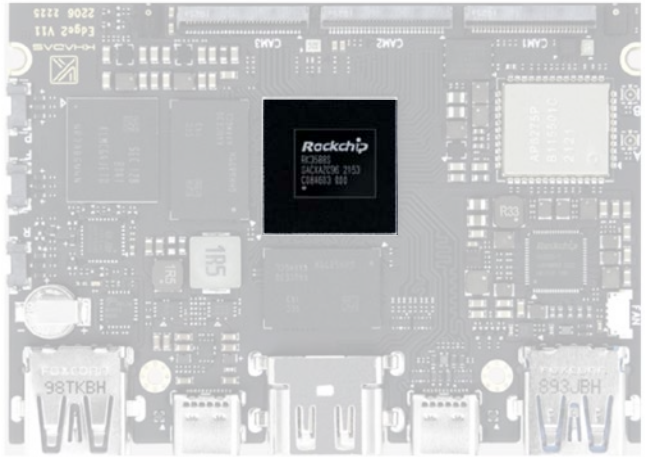
Fuzzer Implantation



Fuzzer Implantation



Supporting Both Discrete and Integrated GPUs



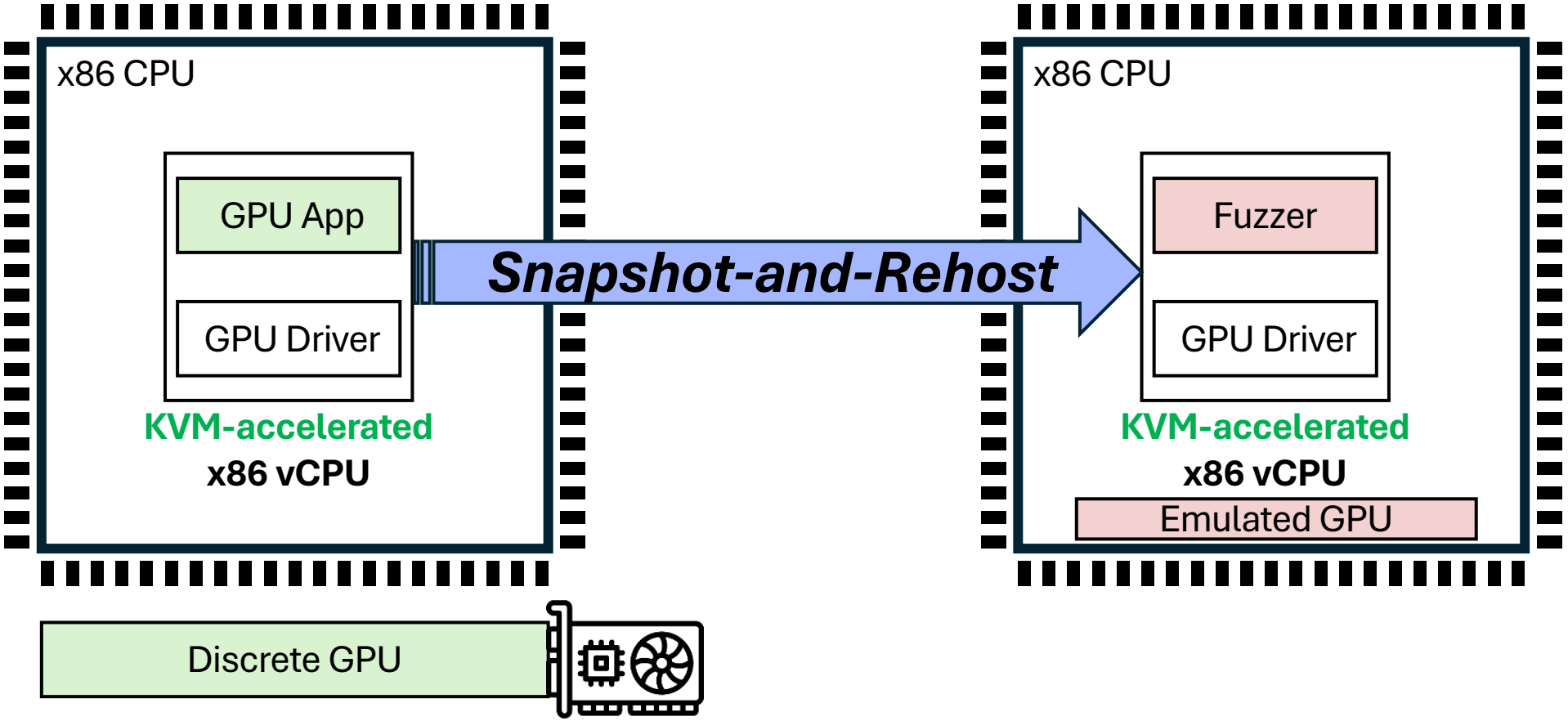
NVIDIA **AMD Radeon**

(Discrete GPUs)

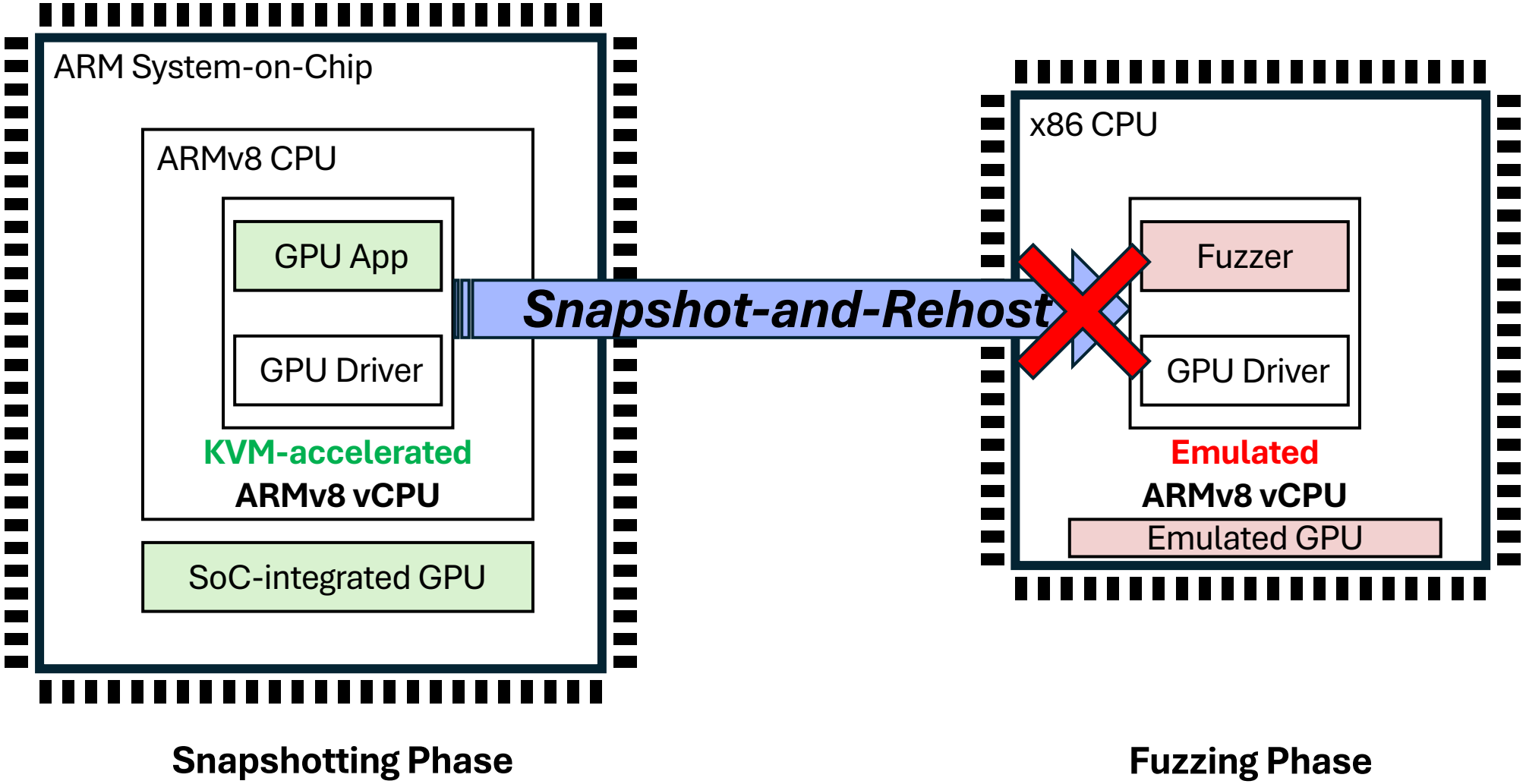
ARM Mali

(Integrated GPU)

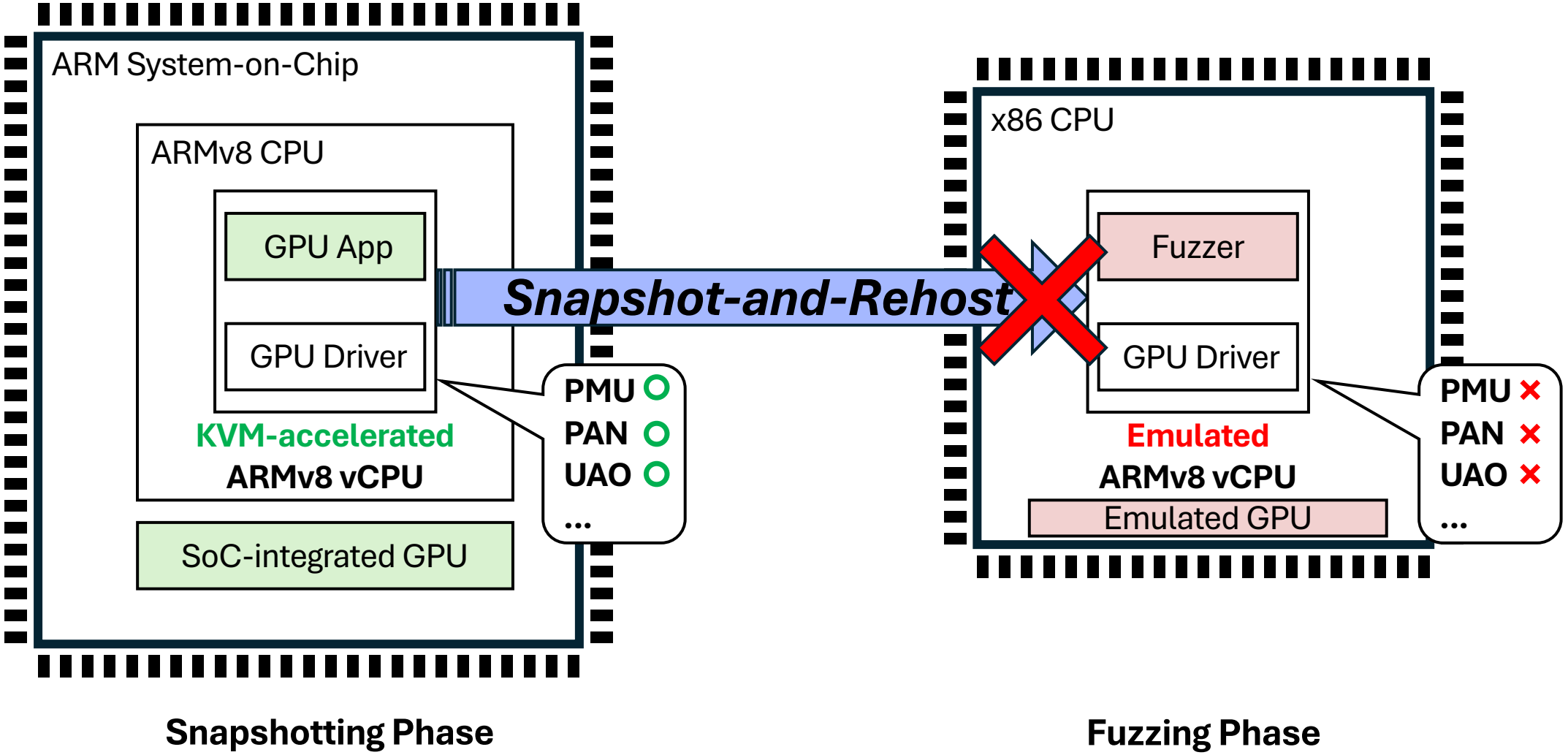
Supporting Fuzzing on x86 Machines



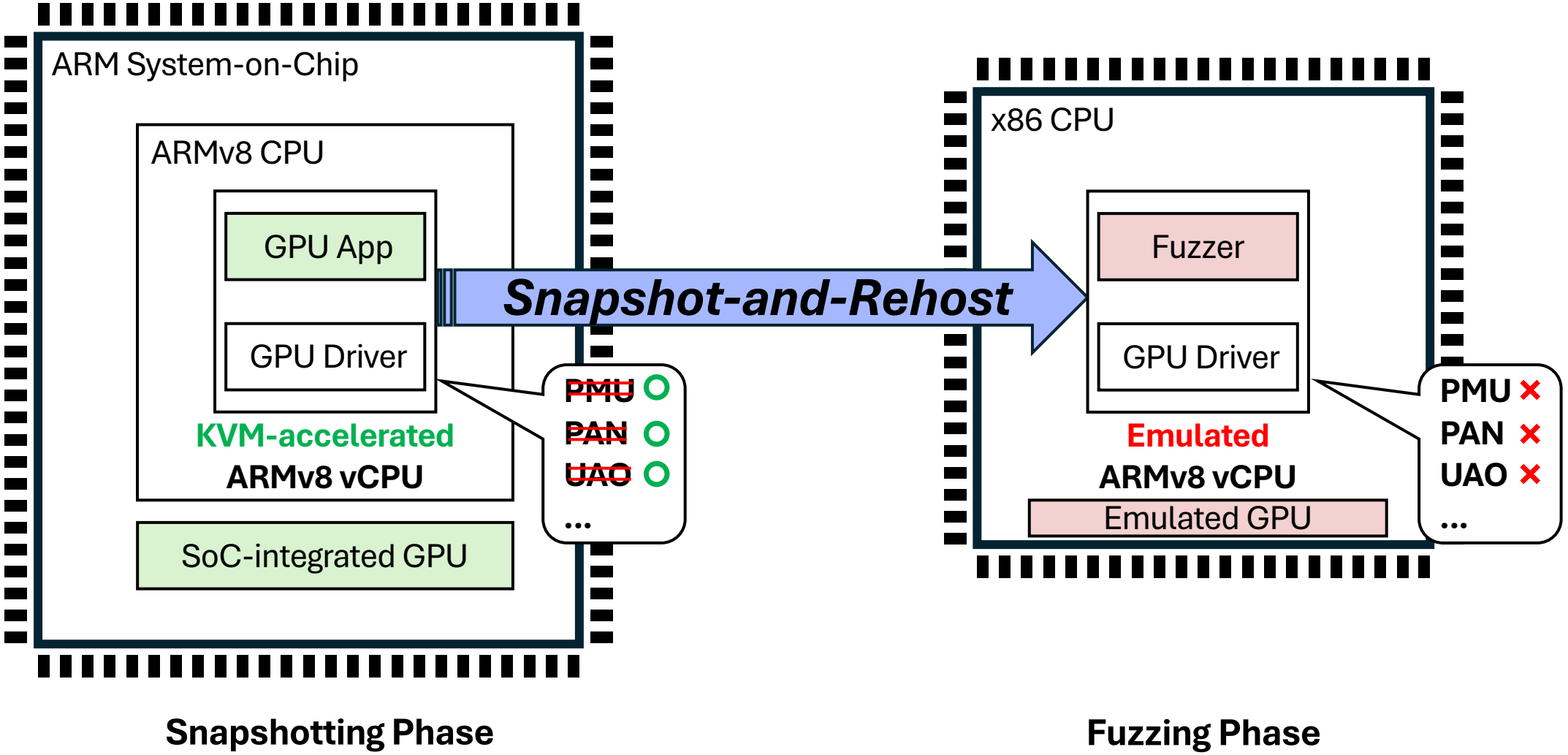
Supporting Fuzzing on x86 Machines



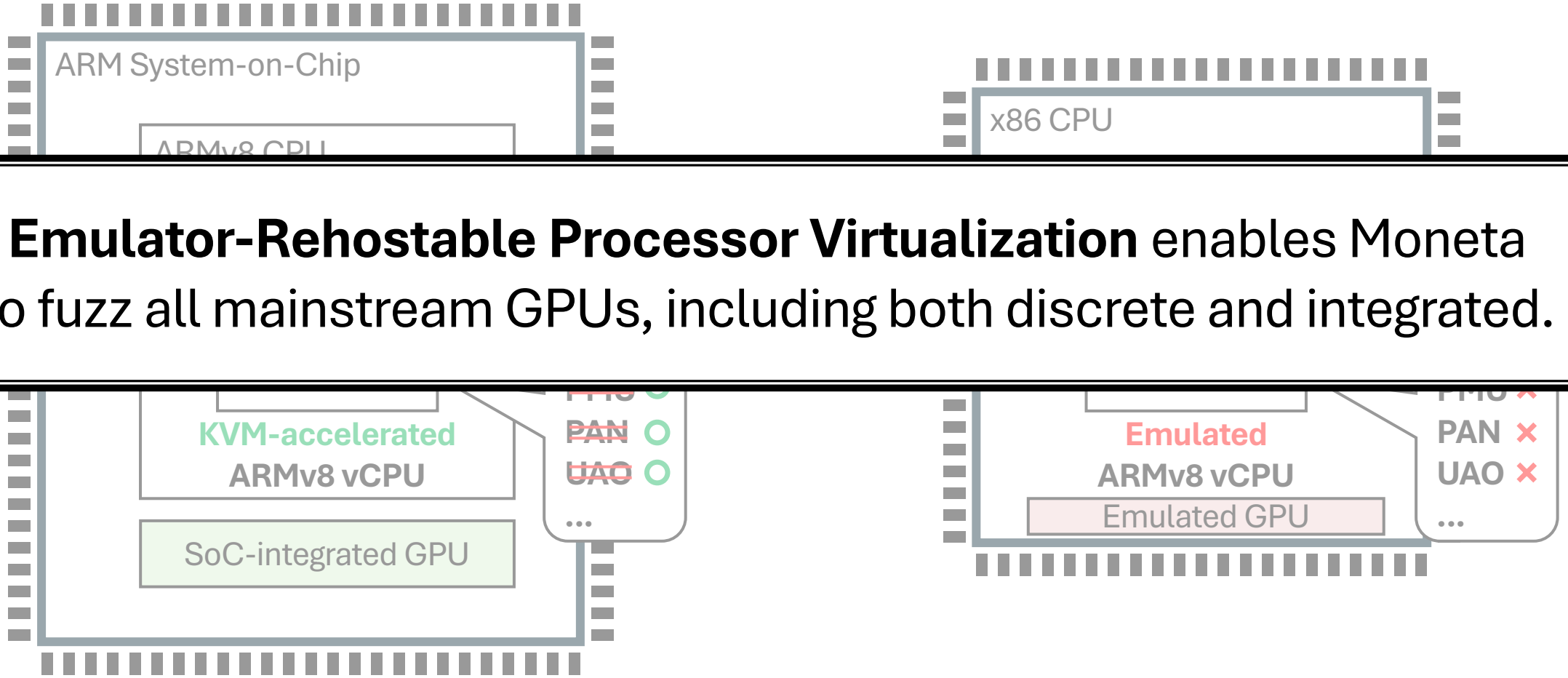
Emulator-Rehostable Virtualization for SoC-Integrated GPUs



Emulator-Rehostable Virtualization for SoC-Integrated GPUs



Emulator-Rehostable Virtualization for SoC-Integrated GPUs



Emulator-Rehostable Processor Virtualization enables Moneta to fuzz all mainstream GPUs, including both discrete and integrated.

Implementation and Experiments

- Implementation of Moneta
  - QEMU 4.0.0 with Linux KVM on x86-64 & AArch64
  - Syzkaller
  - Agamotto[7]

- Experimental Parameters
  - 128 instance parallel fuzzing on Intel Xeon 835
  - 24hr * 3 runs for each experiments
  - WebGL and OpenCL workloads

[7] Song et al., "Agamotto: Accelerating kernel driver fuzzing with lightweight virtual machine checkpoints." *USENIX Security 20*.

GPU Configurations

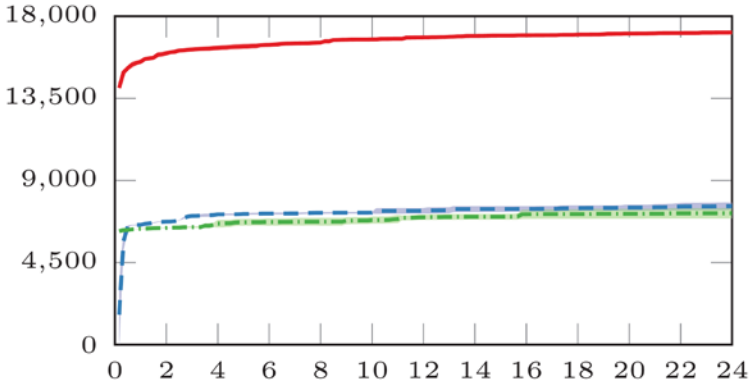
Kernel-Mode GPU Driver	Physical GPU	Kernel-Mode Driver Source Code
NVIDIA	GTX 1650 Super	Official, out-of-tree module[1]
	RTX 3060	
	RTX 4060 Ti	
AMD Radeon	Radeon RX 580	`drivers/gpu/drm/amd/amdgpu` (Upstream)
ARM Mali	Mali-G610 MP4 (on RK3588s)	`drivers/gpu/arm/bifrost` (Downstream[2])

1. https://github.com/NVIDIA/open-gpu-kernel-modules
2. https://github.com/khadas/linux

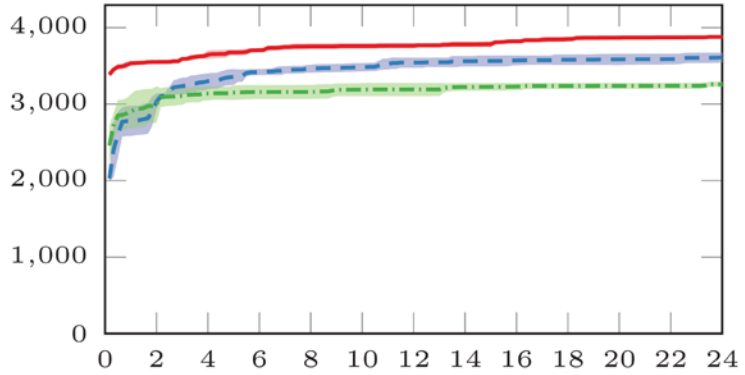
Basic Block Coverage Results

1 Moneta 2 Moneta: No Snapshot 3 Moneta: No Snapshot/Replay

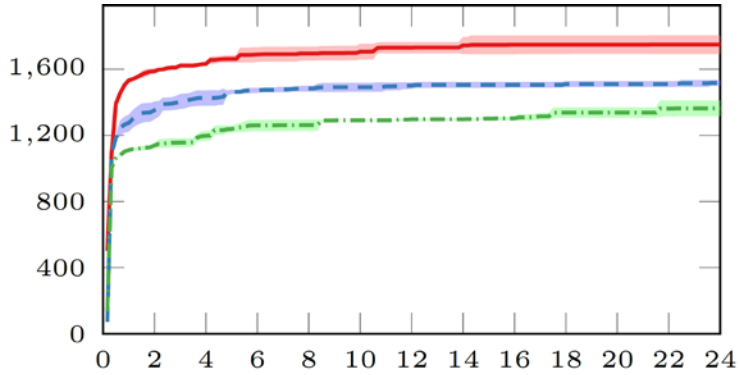
Coverage (Basic Blocks)



NVIDIA GPU Driver



AMD Radeon GPU driver



ARM Mali GPU Driver

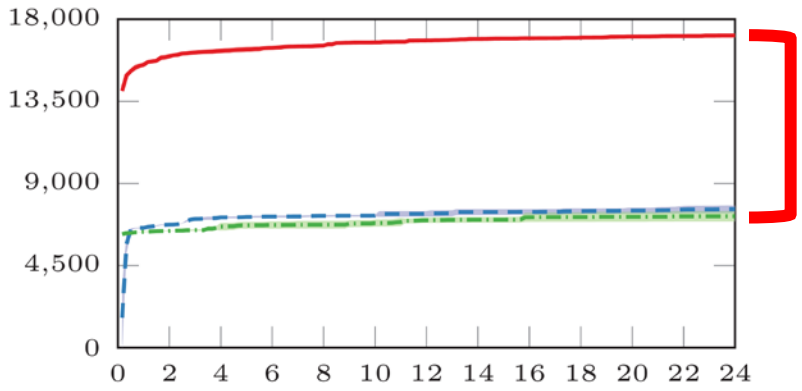
Fuzzing time (Hours)

Basic Block Coverage Results

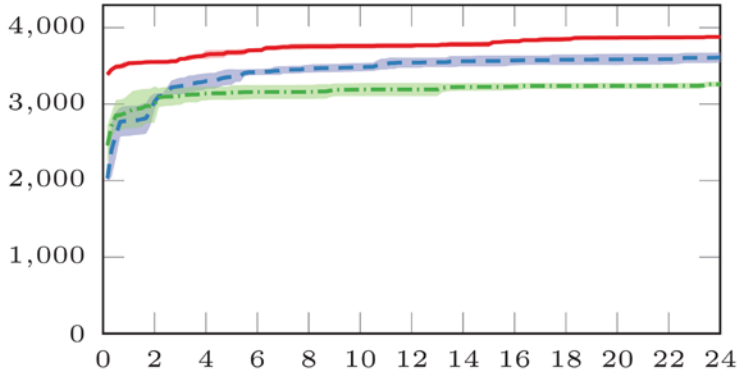
1 Moneta 2 Moneta: No Snapshot 3 Moneta: No Snapshot/Replay

Coverage (Basic Blocks)

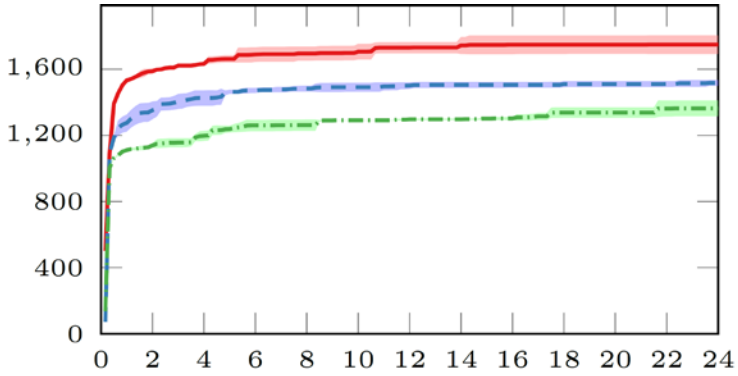
137.8% increase



NVIDIA GPU Driver



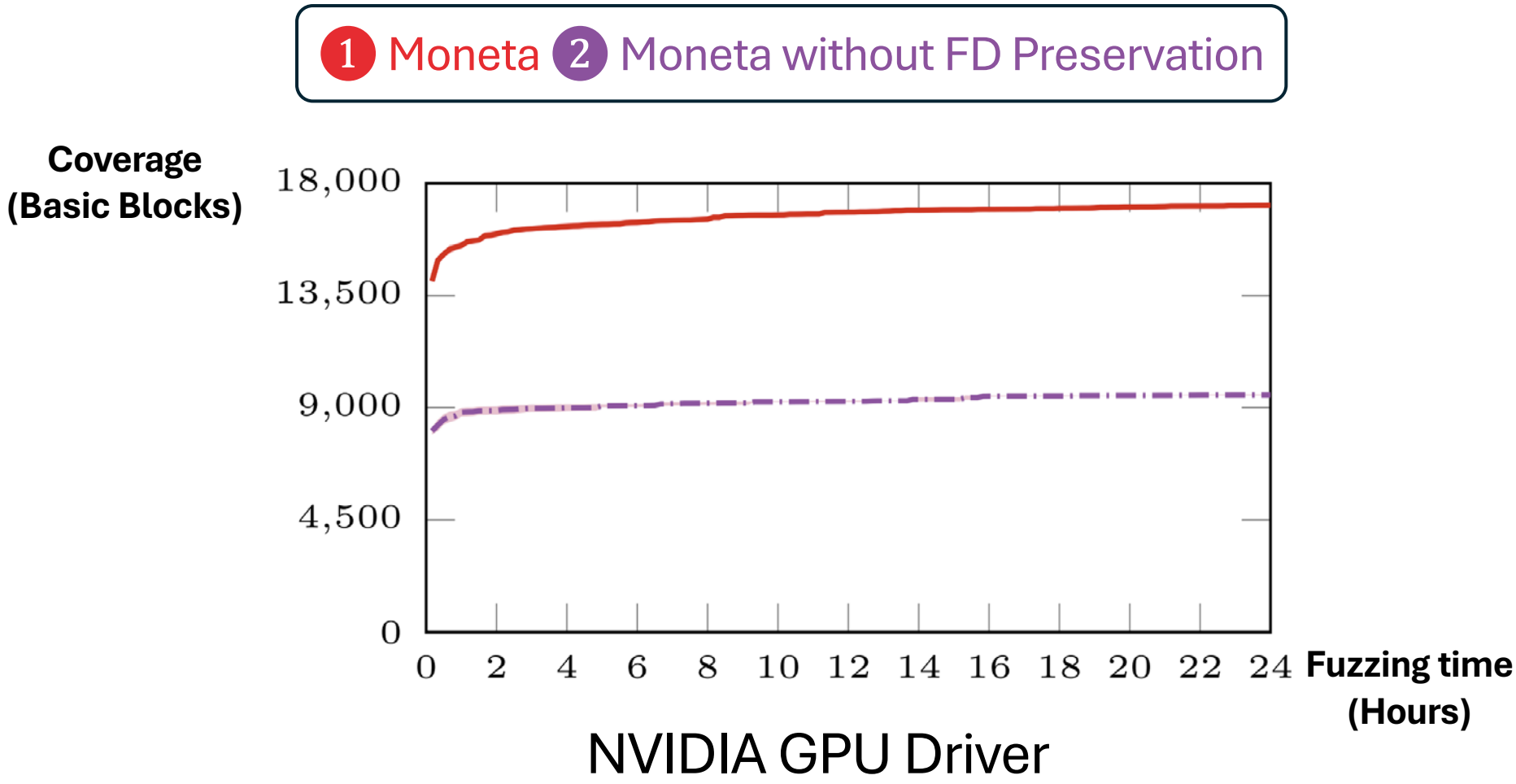
AMD Radeon GPU driver



ARM Mali GPU Driver

Fuzzing time (Hours)

Ablating Moneta's FD Preservation



NVIDIA GPU Driver

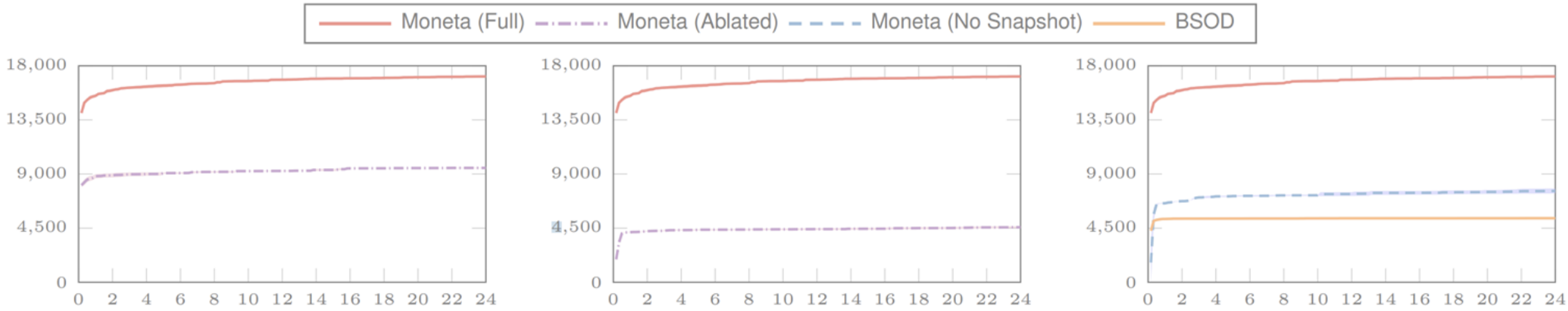
10 Previously Unknown Bugs

GPU Driver	Bug Type	Status
NVIDIA	Slab-out-of-bounds write	CVE-2024-0090
	Slab-out-of-bounds write	
	Slab-out-of-bounds read	CVE-2024-0075[1]
	Null pointer dereference	
	Null pointer dereference	
AMD Radeon	Use-after-free	CVE-2024-26656
	Slab-use-after-free	CVE-2024-27400[1]
	Null pointer dereference	CVE-2024-26657
ARM Mali	Shift out-of-bounds	Confirmed
	Shift out-of-bounds	Confirmed

1. These bugs were concurrently found by the respective vendor or other researchers.

Fuzzing Throughput

GPU Driver	Moneta Configuration	Exec/Minute	Total BB Coverage
NVIDIA	Full	**35,265**	**17,115**
	No Snapshot	21,297	7,588
	No Snapshot/Replay	26,913	7,198
AMD Radeon	Full	19,949	**3,879**
	No Snapshot	17,698	3,612
	No Snapshot/Replay	**21,586**	3,258
ARM Mali	Full	2,332	**1,748**
	No Snapshot	2,738	1,518
	No Snapshot/Replay	**2,995**	1,363



More evaluation results available in the paper.
(ablation studies, bug case studies, etc.)

1 **Execute the workload up to 339 syscalls.**

```
r1 = fetch_preserved_fd$nvidia()
ioctl$nvidia(r1, 0xc0b8464a,
↪ &(0x7f0000000140)="0200d0c10100005c020000000500005c0000000000000000000000000000000d0c1070005c00000000
95c4030300110003000000000000009e6000d87d1eced0000000000000000000000000008000fb000004000000000000000000
0040500000000000000000000000000000000000000000000008aea1713000000000000000000000000400000000000000000
0000000000000000000cd41000000000000000000000000000000000000000000000faff00")
```

2 **Execute the workload up to 9,441 syscalls.**

```
r1 = fetch_preserved_fd$nvidia()
ioctl$nvidia(r1, 0xc0b8464a,
↪ &(0x7f00001f0c00)="0a00d0c10300efbe0600000000000000000000000020000000000000000000000000000000000000
004c4732000000000000000dc00300004000004ffffff23824e9e030084e2f00000000ff0300000000000000000201e0000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000400000000000000000000
0000000000000000000000000000000000002000")
```

3 **Execute the workload up to 341 syscalls.**

```
r1 = fetch_preserved_fd$nvidia()
ioctl$nvidia(r1, 0xc020462a, &(0x7f00001f0c00)="0200d0c10100005c021480000000000bd2a24fbfd7f0000030300")
```

4 **Execute the workload up to 341 syscalls.**

```
r1 = fetch_preserved_fd$nvidia()
ioctl$nvidia(r1, 0xc0384657, &(0x7f000001e600)="0a00d0c10300efbe2300f0ca190000000000000000000000000000200")
ioctl$nvidia(r1, 0xc020462a,
↪ &(0x7f000001e640)="0a00d0c10400efbe1001802000000000097e97d6c7f000084000000000000000")
```

5

```
mknodat$nvidia(0xffffffffffffff9c, &(0x7f0000069200)='/dev/nvidiactl\x00', 0x21b6, 0xc3ff)
r1 = openat$nvidia(0xffffffffffffff9c, &(0x7f0000069200)='/dev/nvidiactl\x00', 0x80802)
ioctl$nvidia(r1, 0xc020462b, &(0x7f0000069c40))
```

GPU Driver	Moneta Configuration	Total Executions (# of Execs/Minute)	Corpus Size (Total BB Coverage)
NVIDIA	Full	**50,781,942 (35,265)**	**1,378 (17,115)**
	No Snapshot	30,667,828 (21,297)	334 (7,588)
	No Snapshot/Replay	38,754,625 (26,913)	262 (7,198)
AMD Radeon	Full	28,726,479 (19,949)	**690 (3,879)**
	No Snapshot	25,485,670 (17,698)	508 (3,612)
	No Snapshot/Replay	**31,084,091 (21,586)**	452 (3,258)
ARM Mali	Full	3,358,603 (2,332)	**235 (1,748)**
	No Snapshot	3,942,659 (2,738)	209 (1,…

Conclusion

- We combine SnR and RnR to design an ex-vivo fuzzing system that **deterministically recalls** deep driver states while still being capable of **evolutionary fuzzing**.

- We implement this idea on a prototype system Moneta, which brought max. **137.8% increase** in block coverage compared to state-of-the-art fuzzers.

- We open-sourced Moneta, which found 10 previously unknown bugs with 5 CVEs assigned.

Q&A

Contact: Joonkyo Jung
joonkyoj@yonsei.ac.kr

Open-source repo: https://github.com/yonsei-sslab/moneta