

Extending Memory State Preservation Across Warm Reboots Using KHO/LUO/memfd

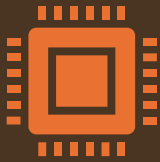
Prasanna Kumar T S M and Srivatsa Bhat
Microsoft Linux Systems Group

LIVE UPDATE MICROCONFERENCE TRACK

**Linux Plumbers Conference,
Prague, Czechia / Oct 6, 2026**

Why can't we always use kexec?

Firmware Updates



Device-Tree Updates



Hardware Errata



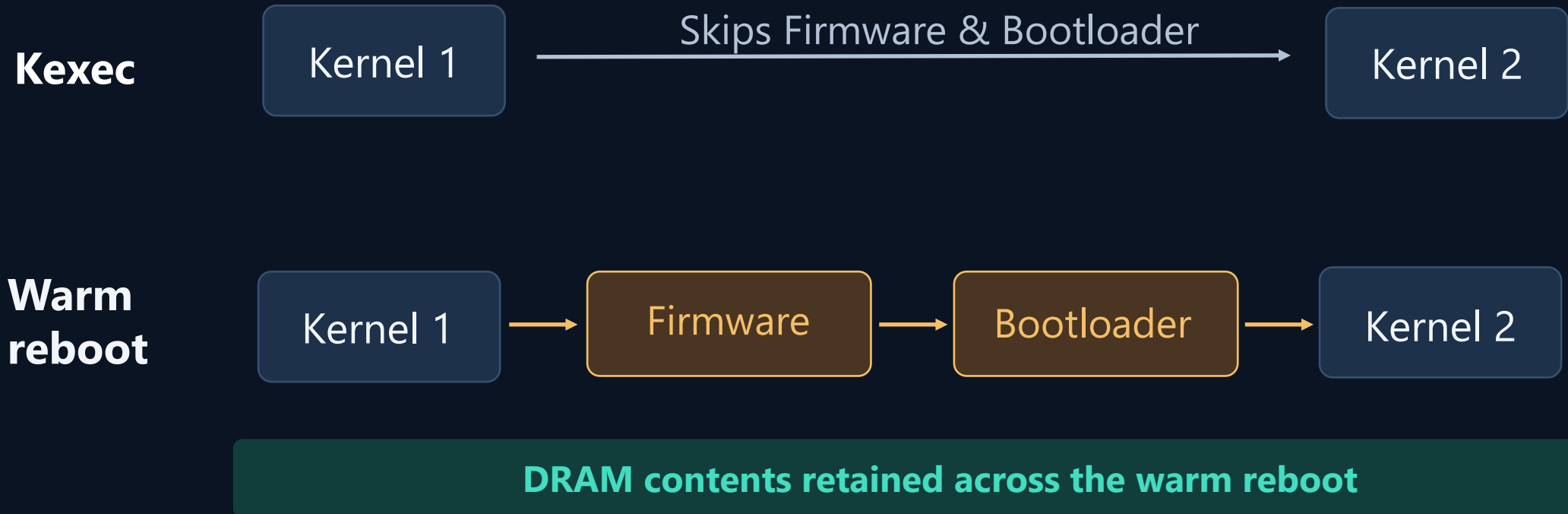
These scenarios require a deeper form of reboot than a kexec – a WARM reboot

GOAL: Extend KHO and its memory state preservation (memfd) to warm reboots, to minimize downtime during these updates.

Background on Target Platform:

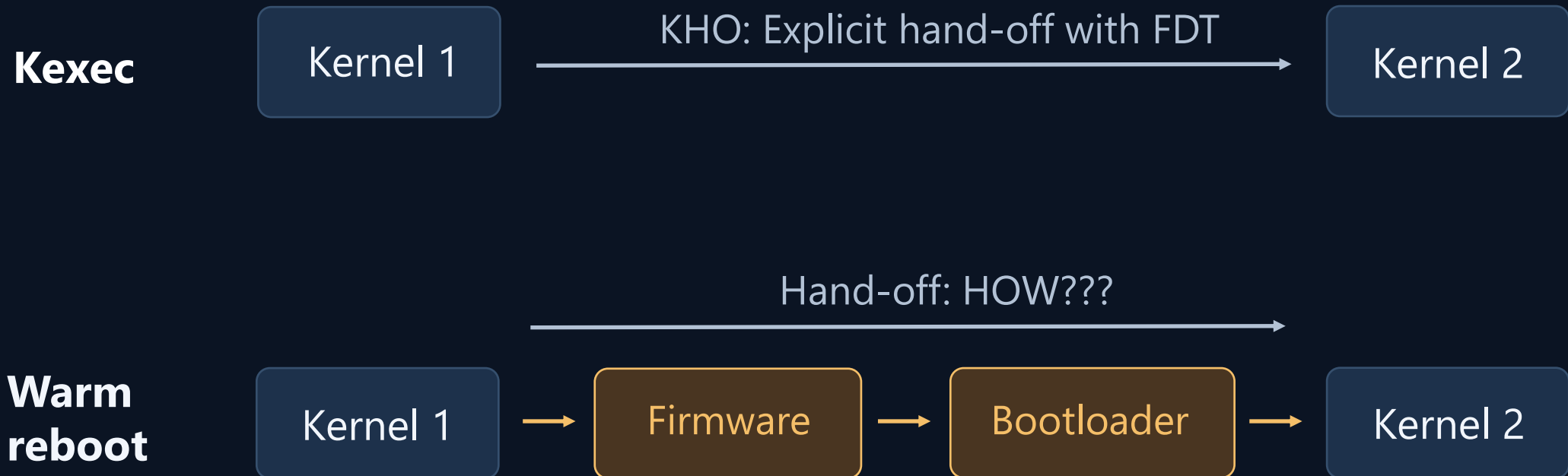
- Bare-metal embedded system
- App state is preserved in DRAM to avoid costly recompute post reboot

What is a warm reboot?



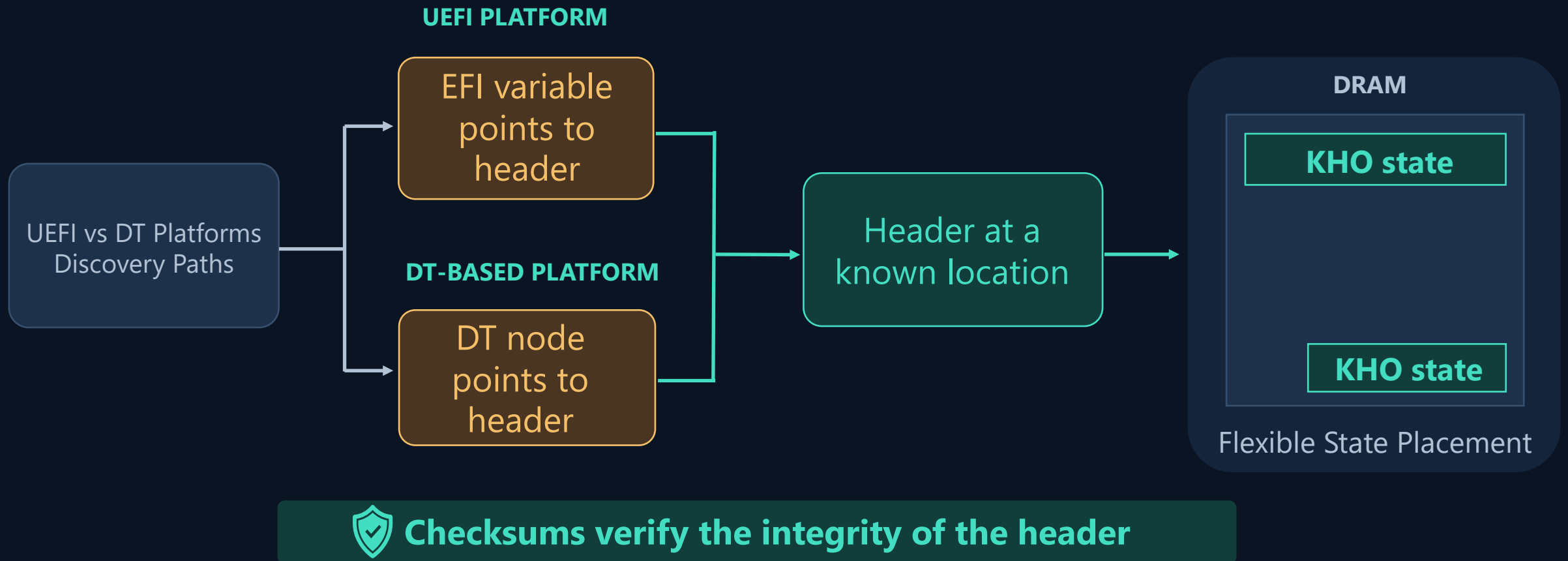
Cold reboot: DRAM is also cleared.

Challenge 1: Discovering preserved KHO state without an FDT



Challenge 1: Discovering preserved KHO state without an FDT

Proposed Solution: Add a level of indirection



Challenge 2: FW can clobber preserved KHO state

- A warm reboot re-runs the firmware and bootloader.

Challenge 2: FW can clobber preserved KHO state

- A warm reboot re-runs the firmware and bootloader.
- "FW-Permanent" Memory: Memory used by FW at runtime; No overlap with Linux

Challenge 2: FW can clobber preserved KHO state

- A warm reboot re-runs the firmware and bootloader.
- "FW-Permanent" Memory: Memory used by FW at runtime; No overlap with Linux
- "**FW-Transient**" Memory: Memory that the FW & Bootloader *use temporarily* only during the warm reboot transition.

Challenge 2: FW can clobber preserved KHO state

- A warm reboot re-runs the firmware and bootloader.
- "FW-Permanent" Memory: Memory used by FW at runtime; No overlap with Linux
- "**FW-Transient**" Memory: Memory that the FW & Bootloader use temporarily only during the warm reboot transition.
- "**FW-Transient**" intentionally overlaps with Linux's **usable-memory** to avoid wasted permanent carveouts.

Challenge 2: FW can clobber preserved KHO state

- A warm reboot re-runs the firmware and bootloader.
- "FW-Permanent" Memory: Memory used by FW at runtime; No overlap with Linux
- "**FW-Transient**" Memory: Memory that the FW & Bootloader use temporarily only during the warm reboot transition.
- "**FW-Transient**" intentionally overlaps with Linux's **usable-memory** to avoid wasted permanent carveouts.
 - SAFETY PROPERTY: FW & Kernel use the same memory at **different** times (i.e., temporal mutual exclusion).

Challenge 2: FW can clobber preserved KHO state

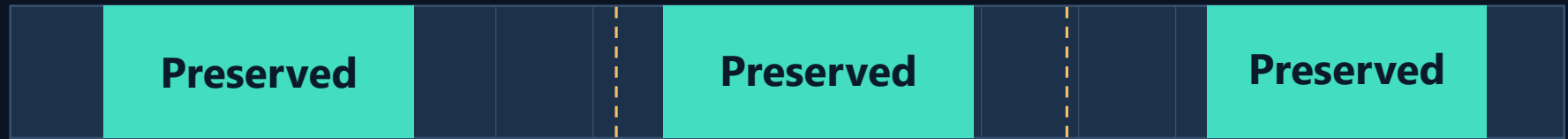
- A warm reboot re-runs the firmware and bootloader.
- "FW-Permanent" Memory: Memory used by FW at runtime; No overlap with Linux
- "**FW-Transient**" Memory: Memory that the FW & Bootloader use temporarily only during the warm reboot transition.
- "**FW-Transient**" intentionally overlaps with Linux's **usable-memory** to avoid wasted permanent carveouts.
 - SAFETY PROPERTY: FW & Kernel use the same memory at **different** times (i.e., temporal mutual exclusion).
 - **This safety property is sufficient for warm reboot, but not for KHO.**

Challenge 2: FW can clobber preserved KHO state

FW-Transient memory overlaps with Linux's usable-memory

THE SAME LINUX-USABLE DRAM ADDRESSES

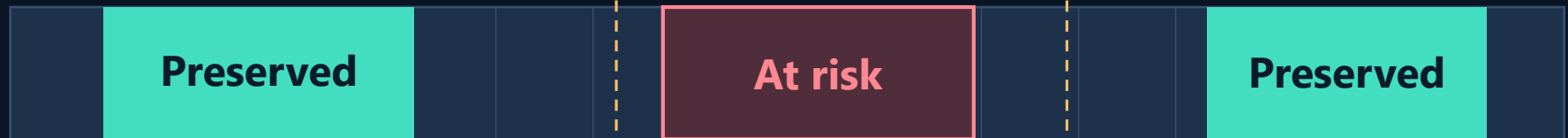
Kernel 1
running



Firmware
transition



Kernel 2
starts



Firmware can clobber state the kernel intended to preserve!

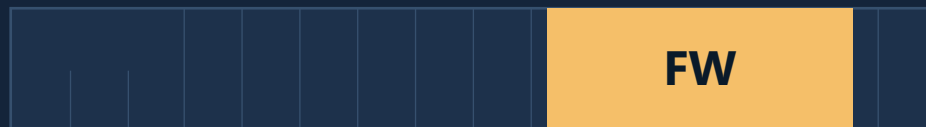
FW-Transient Memory: Fixed at design time per platform

FW-Transient boundaries

Platform A



Platform B



Pre-determined at platform design time

Firmware's fixed memory map

How should the FW-transient boundary be exported to Linux?

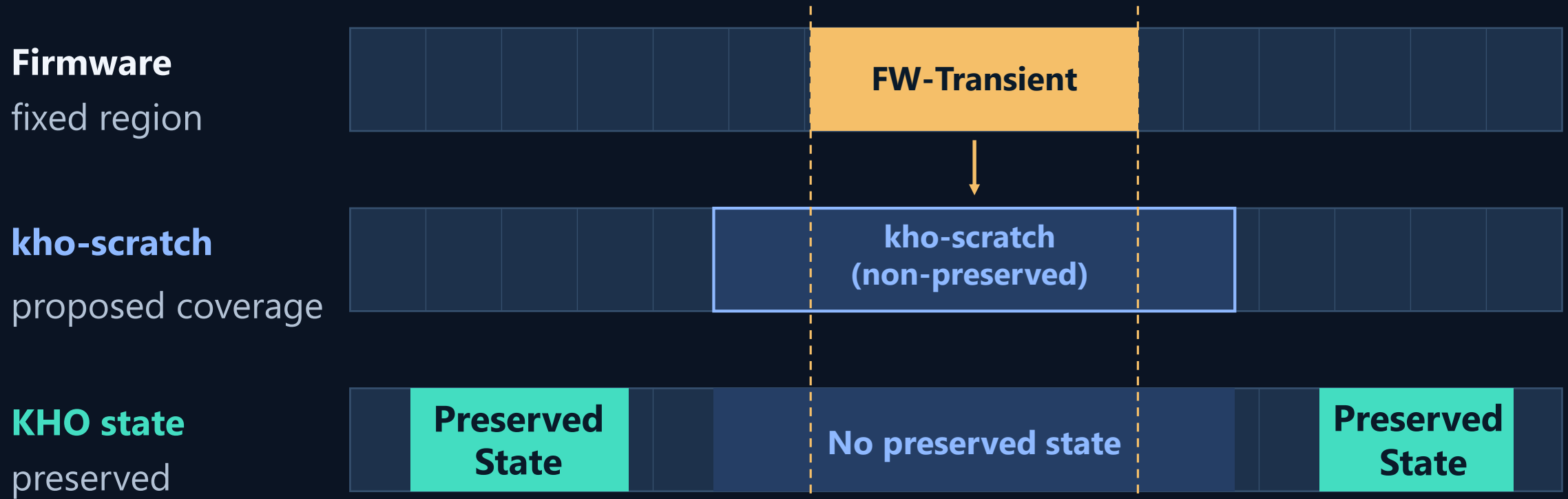
UEFI ?

DT ?

Linux / KHO

Key Insight: FW-Transient cannot move, but KHO can adapt its placement

Make kho-scratch cover FW-Transient

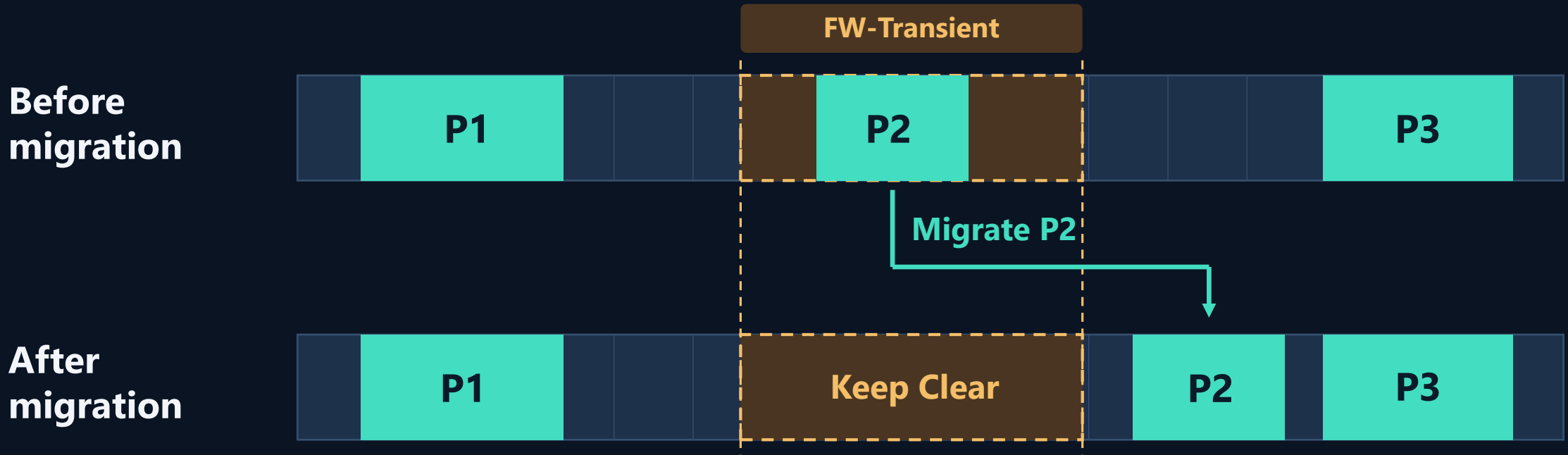


Advantage: Natural fit, as kho-scratch is meant for non-preserved allocations

Catch: Do MM constraints prevent arbitrary sizing & placement of kho-scratch in this way?

Move preserved state from FW-Transient before shutdown

If kho-scratch coverage is not feasible, migrate preserved state out of FW-Transient.



Hard to maintain KHO metadata integrity after migration



Cost on the latency-sensitive reboot path

Preserving memory state safely across a warm reboot

Kernel + Firmware + Bootloader perspectives

CHALLENGE 2 | FW-TRANSIENT DISCOVERY

1. How should FW demarcate the FW-Transient region?
2. How should we expose FW-Transient boundaries on UEFI & DT platforms?

CHALLENGE 2 | MEMORY COLLISION

1. Which constraints determine if kxo-scratch can cover FW-Transient?
2. If migration is inevitable, how can we maintain state integrity without costly reboot work?

Please reach out to collaborate!

- **Prasanna Kumar T S M** <ptsm@linux.microsoft.com>
- **Srivatsa S. Bhat (Microsoft)** <srivatsa@csail.mit.edu>

Microsoft Linux Systems Group